

MINI PROJECT REPORT

on

HEALTHSPACE – Your personal healthcare companion

Submitted in partial fulfilment for the completion of

B.E., IV Semester

INFORMATION TECHNOLOGY

By

HUMA HUSSAIN (160119737067)

ISHIKA GUPTA (160119737068)

Under the guidance of

Mr. Srikanth

Assistant Professor,

Dept. of IT, CBIT.



DEPARTMENT OF INFORMATION TECHNOLOGY

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

(Affiliated to Osmania University; Accredited by NBA(AICTE) and NAAC(UGC), ISO

Certified 9001:2015)

GANDIPET, HYDERABAD – 500 075

Website: www.cbit.ac.in

2020-2021



**CHAITANYA BHARATHI
INSTITUTE OF TECHNOLOGY (A)**
Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in

Affiliated to

Re-accredited by
 NAAC
for all Programs

100th Rank in
 NIRF
COPR- 311 Grade A

ISO
9001:2015

COMMITTED TO
RESEARCH,
INNOVATION AND
EDUCATION

41
years

CERTIFICATE

This is to certify that the Mini Project-II entitled "**HEALTHSPACE**" submitted to **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**, in partial fulfilment of the requirements for the completion of B.E., IV semester Information Technology, during the academic year 2020-2021, is a record of original work done by **HUMA HUSSAIN (160119737067) & ISHIKA GUPTA (160119737068)** during the period of study in the Department of IT, CBIT, HYDERABAD, under my supervision and guidance.

Project Guide

Mr. Srikanth

Professor, Dept. of IT,
CBIT, Hyderabad.

Head of the Department

Dr.K Radhika

Professor & Head, Dept of IT,
CBIT, Hyderabad.

DECLARATION

We the undersigned solemnly declare that the project report is based on our own work carried out during the course of our study under the supervision of Mr. Srikanth.

We certify that:-

- The portrayal of the desktop application is original and has been done by us under the general supervision of our supervisor.
- The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.
- We have followed the guidelines provided by the university in writing the report.
- Whenever we have used materials (data, theoretical analysis, and text) from other sources, we have given due credit to them in the text of the report and giving their details in the references.

Huma Hussain (160119737067)

Ishika Gupta (160119737068)

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our teacher Mr. Srikanth who gave us the golden opportunity to do this wonderful mini project titled ‘Healthspace’ which also helped us in doing a lot of research and assisted us in expanding our programming skillsets and creative thought process too.

Secondly, we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

Lastly, a special appreciation to our partners for the constant support we’ve had for each other and the learning we did together.

ABSTRACT

The health system is an essential socio-economic activity and therefore it requires rational and effective management. An efficient hospital management system can go a long way and can make administrative work in hospitals, which is just as important as any other hospital work, easy.

Hospitals act as the main actors of the healthcare systems and consequently generate an essential volume of information. Unfortunately, in most cases, this vital information is dispersed or is not available in the necessary time and format, leading to disasters.

“Healthspace” makes use of tools such as database management system, Java, and a proper java database connectivity to efficiently manage crucial information about medical records, patients, doctors, and bills, by having a database of each model implemented. By digitizing most of the processes in the institution, “Health Space” improves customer service and reduces the process cost.

In addition, “Healthspace” controls and schedules online appointments in such a way that the patient is allowed to look at the information and qualifications of the doctors available before actually booking an appointment with them. This increases the customer satisfaction.

Right from entering the patient into the database, into the health service, to finally discharging them, “Healthspace” makes managing information very easy, for the patient as well as the doctor. It also makes it extremely convenient to cancel appointments by the simple click of a button.

“Healthspace” is the go to place for you to manage your health in the easiest way possible.

TABLE OF CONTENTS

TITLE	PAGE NO.
Abstract	V
List of Figures	VII
 1. INTRODUCTION	 X
1.1 Introduction	X
1.2 Application	X
1.3 Motivation	XI
1.4 Problem Statement	XI
1.5 Aim and objective	XI
 2. SYSTEM REQUIREMENTS	 XII
2.1 Functional requirements	XII
2.2 Non-functional requirements	XIV
2.3 Software requirements	XIV
2.4 Hardware requirements	XV
 3. METHODOLOGY	 XVI
3.1 Architecture	XVI
3.1.1 ER diagram	XVI
3.1.2 JDBC classes	XVII
3.2 Assumptions	XVIII
3.3 Requirements	XVIII
 4. IMPLEMENTATION, TESTING AND RESULTS	 XIX
 5. CONCLUSION AND FUTURE SCOPE	 LIX
5.1 Conclusion	LIX
5.2 Limitations	LIX
5.3 Future Scope	LIX
 6. BIBLIOGRAPHY	 LX

LIST OF FIGURES

Figure Name	Page Number
2.1 Java Interpreter	xii
2.2 Architecture of JDBC4.1	xiii
4.2 ER Model	xvi
4.3 Login page	xx
4.4 Testing invalid credentials	xxi
4.5 Login page code snippet	xxi
4.6 Login page code snippet 1	xxii
4.7 Login page code snippet 2	xxii
4.8 Create patient	xxiii
4.9 Patient created	xxiv
4.10 Invalid patient registration	xxiv
4.11 Create patient code snippet	xxv
4.12 Create doctor	xxv
4.13 Doctor created	xxvi
4.14 Create doctor code snippet	xvi
4.15 Reset doctor password	xxvii
4.16 Invalid password change	xviii
4.17 Reset password code snippet 1	xxix
4.18 Reset password code snippet 2	xxix
4.19 Patient login features	xxx
4.20 Patient login code snippet	xxx

4.21 Doctor login features	xxxii
4.22 Doctor login code snippet	xxxiii
4.23 Patient details	xxxiii
4.24 updating patient details	xxxiv
4.25 Patient details code snippet	xxxv
4.26 Doctor details	xxxv
4.27 Doctor details update	xxxvi
4.28 Doctor details code snippet	xxxvii
4.29 View doctors	xxxviii
4.30 View doctors code snippet	xxxix
4.31 List of patients	xxxix
4.32 List of patients code snippet	XL
4.33 Ask patient code snippet	XL
4.34 Update health status	XL
4.35 Updating health status	XL
4.36 Update health status code snippet 1	XLII
4.37 Update health status code snippet 2	XLII
4.38 Patient's appointments	XLIII
4.39 Patient's appointments code snippet	XLIV
4.40 Doctor's appointments	XLIV
4.41 Doctor's appointments code snippet	XLV
4.42 Booking an appointment	XLVI
4.43 Invalid appointment booking	XLVI
4.44 Appointment booking code snippet	XLVII
4.45 Appointment booking code snippet	XLVII
4.46 Fee payment	XLVII
4.47 Cancelling appointment	XLVIII
4.48 Refund after cancellation	XLIX

4.49 Cancelling appointment code snippet	L
4.50 Writing report	LI
4.51 Submitting report	LI
4.52 Writing report code snippet	LII
4.53 Writing prescription	LIII
4.54 Submitting prescription	LIII
4.55 Writing prescription code snippet	LIV
4.56 Viewing report	LIV
4.57 Viewing report code snippet	LV
4.58 Viewing prescription	LVIII

INTRODUCTION

1.1 INTRODUCTION:-

Health is a state of physical, mental, and social well-being in which disease and infirmity are absent. For a smooth lifestyle, a healthy system is of necessity. Moreover, its functioning and management is of further importance, especially for hospitals that are the sole holders of health administration.

For effective and convenient management of various processes that take place in a hospital, hospital management systems come into picture. Hospital management systems allow us the ability to optimize and digitize all the processes within the institution, which will help to improve customer service, reduce process costs, streamline the search of medical records, bills, patients, doctors, etc. They also enable having a record which is specific for each module whenever the database is in need.

All in all, ‘Healthspace’ acts as the most effective interface between doctors and patients which is why it truly is your personal healthcare companion!

1.2 APPLICATION:-

‘Healthspace’ has its roots from hospital management systems from where it branches out to not just being a robust health experience but also a catalyst for unhindered interaction between doctors and patients. Its value also notices a surge considering the ongoing pandemic as it facilitates virtual communication. By using tools like database management system, java and JDBC, it successfully stores all critical information, including patient-specific disease, required by a hospital and its clients and displays them accordingly.

Being a user-friendly application, ‘Healthspace’ provides segregated domains for doctors and patients for their information to be stored and accessed easily. Its major feature is to efficiently schedule and cancel appointments whenever the necessity rises. It is an application that will stick by its user right from the registration process and keeps constant track of any updates.

1.3 MOTIVATION:-

A health system is important and therefore it is necessary to have an effective management for hospitals. Hospitals act as the main actors of the healthcare systems and consequently generate an essential volume of information. They're the sole holders of medical administration.

Unfortunately, in most cases, this vital information is dispersed or is not available in the necessary time and format, leading to fatal peccadillos. By collecting and saving patients' information, including diagnosis reports, medical history, allergy reactions, vaccinations, treatment information plans, test results, etc., Health Information Systems provide the healthcare providers a complete and orderly framework that helps them interact with their patients in a better way and eventually deliver care to them in a more efficient way. This is the sole reason that drove us to create a platform that enabled us to do this.

1.4 PROBLEM STATEMENT:-

To create a user friendly and efficient hospital management system facilitating easy booking and cancelling of appointments and better interaction between doctors and patients.

1.5 AIM & OBJECTIVE:-

The main driving force for this project was the ongoing pandemic that has urged people to take a closer look at their health which has increased the necessity and usage of healthcare systems that creates an efficient interface between patients and doctors in these testing times.

SYSTEM REQUIREMENTS

2.1 FUNCTIONAL REQUIREMENTS:-

Java is both a programming language and a platform. You can use it to write or create various types of computer applications. The java platform is a new software platform which is designed to deliver and run highly interactive, dynamic and secure applications on networked computer systems. Programs written in java are compiled into java byte code which is then interpreted by a special java interpreter (JAVA VIRTUAL MACHINE-JVM) for a specific platform. The JVM is an abstract machine designed to be implemented on the top of the existing processors. It hides the underlying operating system from java applications. The JVM can be implemented in software and hardware. JVM combined with Java API (Application Programming Interface) are libraries of compiled code that can be used in programs.

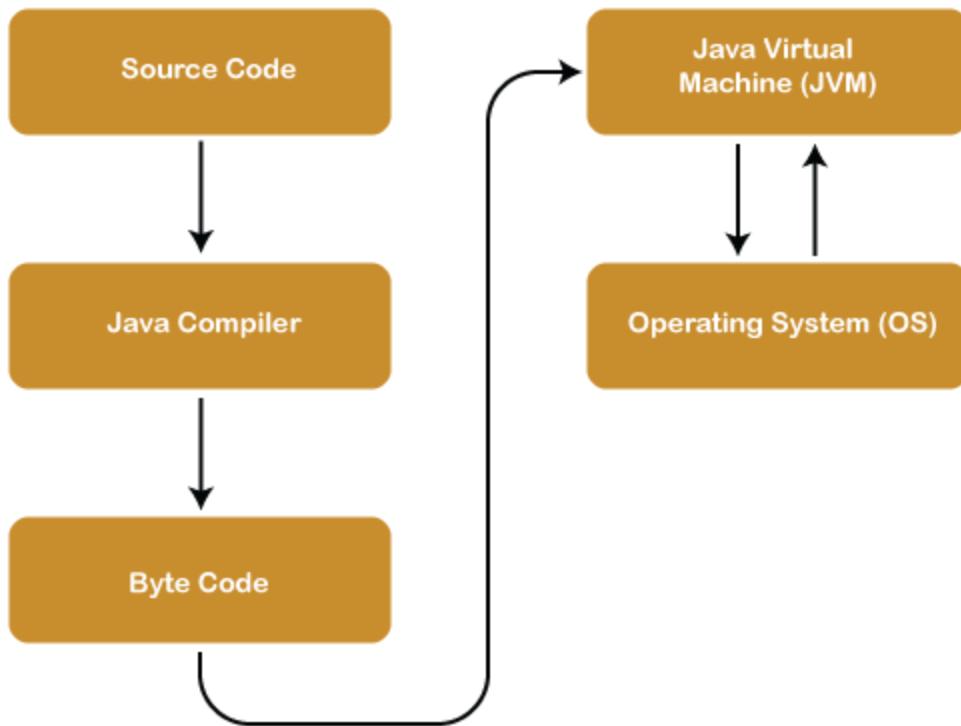


Fig 2.1: Java Interpreter

When we design real life applications such as a hospital database management system, we encounter situations wherein we need to manipulate data stored in a database through the application we have designed.

Since we developed a java GUI application using NetBeans IDE, we have used the concept of JDBC(JAVA DATABASE CONNECTIVITY).

We have connected to a relational database (MySQL) from within a java application using JDBC. JDBC establishes a connection with the MySQL database, sends SQL statements to the database server and processes the results obtained. These various data accessing functions are handled by the JDBC API which consists of a set of Java classes that use predefined methods.

The 4 main classes in the JDBC API include: DriverManager, Connection, Statement, and ResultSet.

Architecture of JDBC

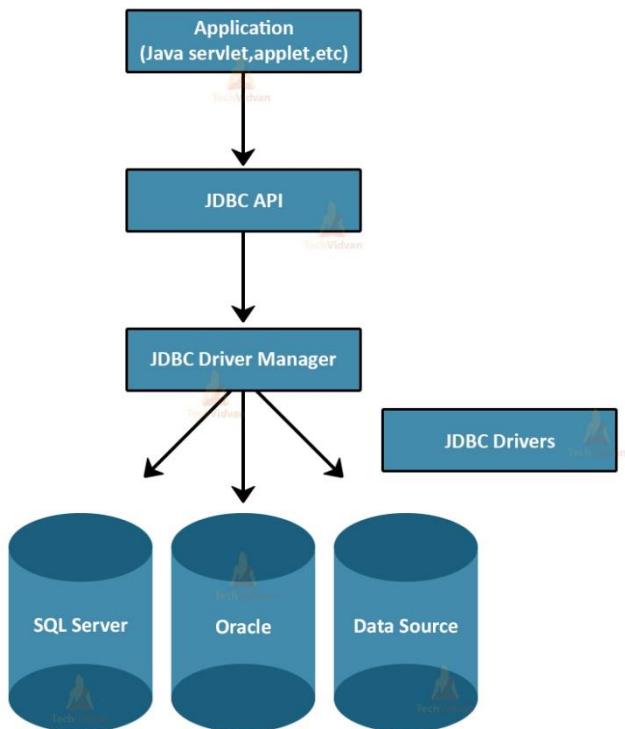


Fig 2.2: Architecture of JDBC

2.2 NON-FUNCTIONAL REQUIREMENTS: -

GUI stands for Graphical User Interface. In java, GUI features are supported via JFC (Java Foundation Classes). One major feature of JFC is Swing API which includes everything from buttons to split panes to tables and every other functionality to create a GUI application. There are 3 tiers of software in GUI programs created using Java Swing:

Graphical components that make up the GUI such as a button or textbox or menu etc. Each graphical component has certain properties. A property is a characteristic of an object such as its size, color, title etc.

Event listeners that get notified about events and respond to them.

Event handler that do the work for the user in case the event occurs.

The graphical controls that you put in a GUI application are broadly of two types: Container and child controls. The container can hold other controls within it (examples: frames, panels).

Controls inside a container are child controls. A GUI application must contain a top level container that can hold other controls.

The three major players that add functionality to GUI are:

The event: it is an object that gets generated when user does something such as mouse click, dragging, pressing a key on the keyboard etc.

The source of the event: the component where the event has occurred is the source of the event.

The event listener: it is attached to a component and contains the methods that will execute in response to the event. These methods are stored inside the listener interface.

2.3 SOFTWARE REQUIREMENTS

NetBeans IDE is a free, open-source, cross-platform IDE(Integrated Development Environment) with built-in support for java programming. It offers a RAD style of developing for Windows programmers. It enables us to develop applications by creating frames, dragging and dropping

controls onto the frames, setting properties for various things and adding some application specific code to handle events

MySQL is a freely available open source Relational Database Management System (RDBMS) that uses Structured Query Language(SQL). In a MySQL database, information is stored in tables. It provides a rich set of features that support a secure environment for storing, maintaining and accessing data. It is a fast, reliable and scalable alternative to many of the commercial RDBMSs available today. It has a client-server architecture and is a multiuser database system.

2.4 HARDWARE REQUIREMENTS:-

When it comes to hardware needed, it is very simple. To create this desktop application, we needed a computer. It is easy to create this by using Windows PC and Windows 7 as these operating systems are stable and reliable. Another consideration for hardware is some sort of backup hard drive or removable USB memory stick(s) as It is important that we back up all of our website data to a removable drive in the event a computer crashes, is broken or is stolen. If we lose this data, we will be forced to start from scratch in many cases.

METHODOLOGY

3.1 ARCHITECTURE:-

3.1.1 ER DIAGRAM

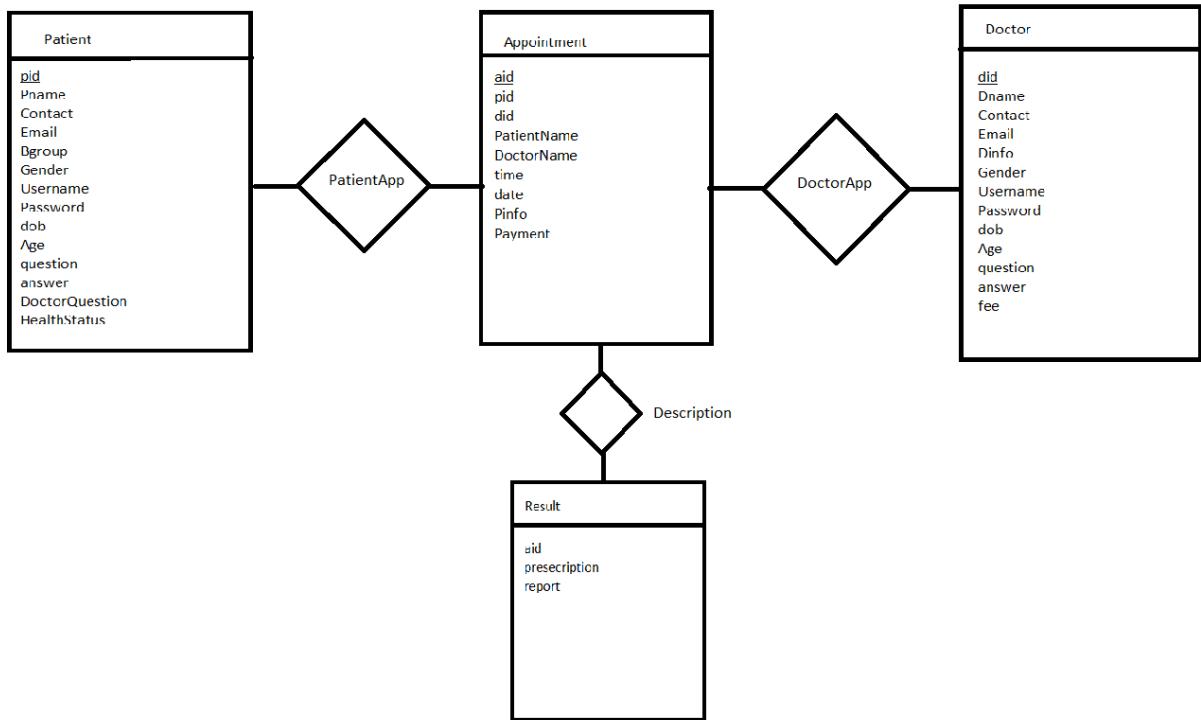


Fig 3.1: ER Model

The various tables included in our database include: patient , doctor , appointment , and result.

The patient table holds all information with respect to the patient. This includes their name, contact, email, blood group, gender, username, password, date of birth, age, security question, its answer, question asked by the doctor, and your answer to it. Upon registering a patient, each patient will be given a unique identifying number which is the pid attribute. This acts as the primary key of the table. Similarly, we have the doctor table which hold all relevant information with respect to the doctor. This includes their name, contact, email, specialization, gender, username, password, date of birth, age, security question, its answer, and the fee per appointment. The did attribute acts as the primary key of the table.

The appointment table keeps track of the appointments booked by various patients with various doctors. Every appointment is given a unique identification number which acts as the primary key of the table. The patient id and doctor id attributes of the table are foreign keys referring to the patient and doctor tables respectively. Apart from this the table contains the patient's name who has booked the appointment, the doctor's name with whom they have booked the appointment, the time and date on which they can take the appointment, the problem they are facing and the payment details.

The result table keeps track of the report and prescription written by the doctor for their patient for a particular appointment. The appointment id acts as the primary key. Prescription and report are attributes of the table.

3.1.2 JDBC CLASSES:

The four main classes in the JDBC API hierarchy that are generally used for the database connectivity are:

Driver Manager Class: it loads the JDBC driver needed for to access a particular data source, locates and logs on to the database and returns a connection object. The required JDBC driver must be registered with the JDBC Driver Manager before a connection between the java application and the JDBC data source can be made. Once a driver is registered and loaded, the driver manager class can use that driver to connect the java application to its associated data source, i.e , dbms. Once the connection is established, all database calls go directly to the JDBC driver itself by passing the Driver Manager.

Connection Class: it manages the communication between a java client application and a specific database (example: MySQL database), including passing SQL statements to the DBMS and managing transactions.

Statement Class: it contains SQL strings that are submitted to the DBMS. An SQL Select statement returns a ResultSet object that contains the data retrieved as the result of SQL statement.

3.2 ASSUMPTIONS:-

The assumptions we have made is that the systems have NetBeans installed in order to run this application.

In addition to that, we assume that the data being stored in the database is authentic and genuine.

3.3 REQUIREMENTS:-

We require that there is a large database for the application to run as efficiently as possible and that the information in the database is genuine.

IMPLEMENTATION, TESTING AND RESULT OF THE PROJECT

With the use of GUI on NetBeans IDE, all the pages were keenly created. The desktop application begins with the login page which comprises several buttons, labels and text fields that have been separately coded to serve their own purpose. All pages contain common header, footer, and center labels to represent unity amongst them all. All pages contain the back button that navigate them to the previous frame.

All the codes given below begin with a connection being made between the code and MySQL built-in classes provided by JDBC. Then a MySQL query is sent to access the data from the database which is then used in the JFrame.

The login page asks the user for their credentials if they have already registered as a doctor or a patient. If not, the login page also contains buttons that direct the user to create an account as a doctor or a patient. Once the user has entered his/her credentials, they can choose the radio button that takes them to their home page. The page also contains buttons that are required whenever a user forgets his/her password. This page comprises constraints that will not allow a patient to login with a doctor's credentials and vice versa. If a username is invalid or not present in the database, a pop-up will be shown which does not allow the user to log in. The page is

efficiently created to cater to doctors and patients separately.

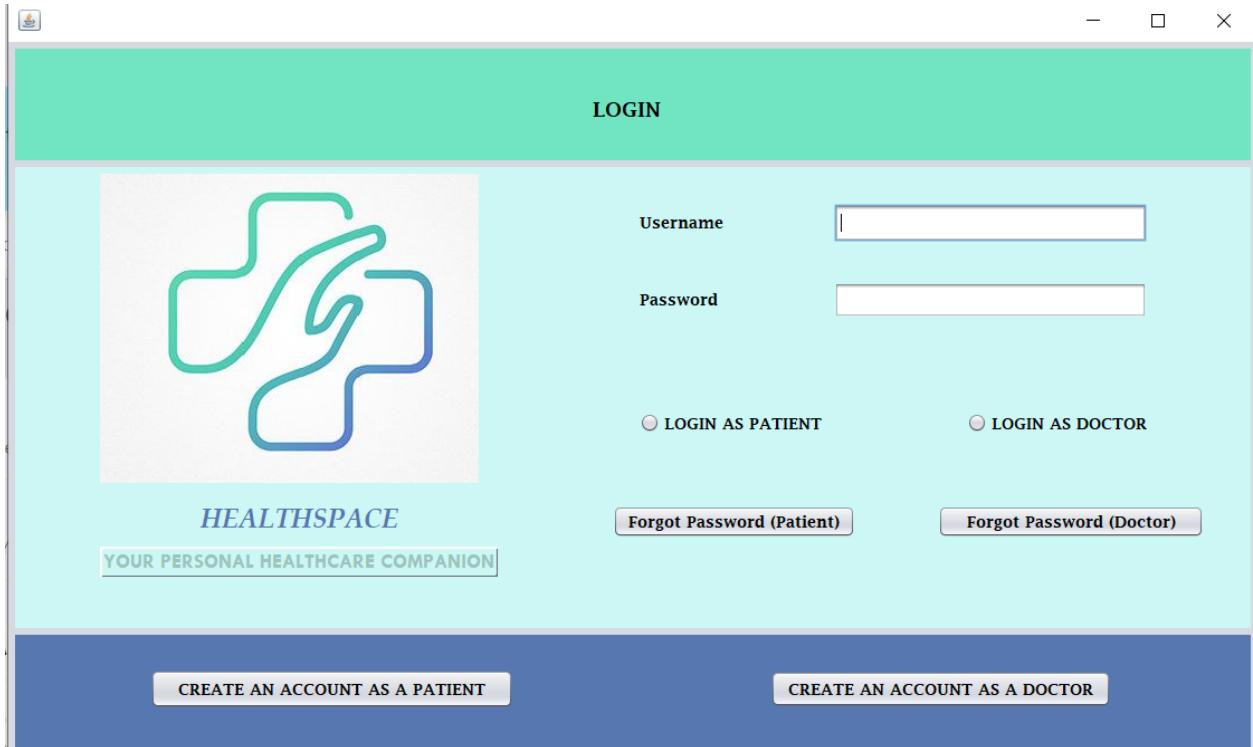


Fig 4.1: Login page

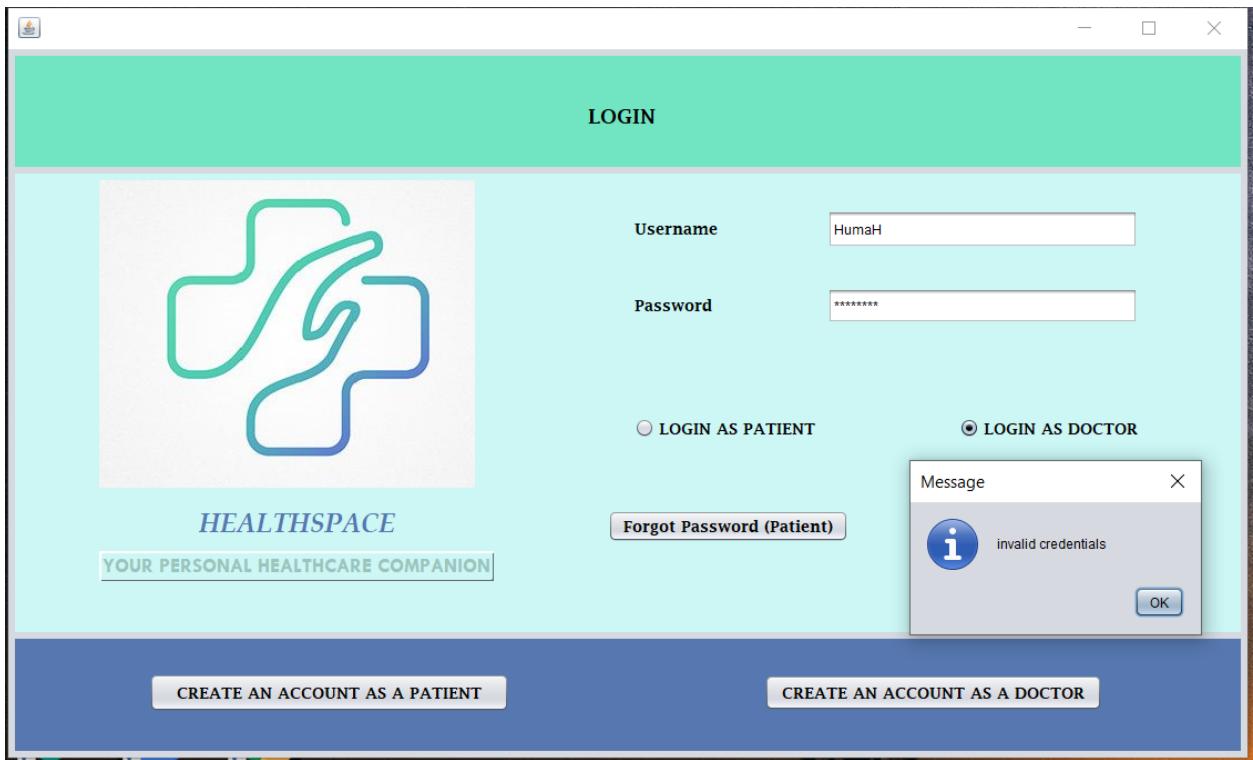


Fig 4.2: Testing invalid credentials

The screenshot shows the NetBeans IDE interface with the project 'Health_space' open. The code editor window displays the 'Home.java' file. The code is as follows:

```

1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6
7
8  /**
9  * Author: Hello
10 */
11
12
13 /**
14 * Creates new form Home
15 */
16
17 public class Home extends javax.swing.JFrame {
18
19     /**
20      * This method is called from within the constructor to initialize the form.
21      * WARNING: Do NOT modify this code. The content of this method is always
22      *
23     public Home() {
24         initComponents();
25         username=jl1.getText();
26         password=jl2.getText();
27     }
28
29
30     /**
31      * This method is called from within the constructor to initialize the form.
32      * WARNING: Do NOT modify this code. The content of this method is always
33      *
34     @Override
35     public void actionPerformed(ActionEvent evt) {
36         if(r1.isSelected()){
37             if(username.equals("HumaH") & password.equals("123456")){
38                 JOptionPane.showMessageDialog(null, "Success");
39             }
40             else{
41                 JOptionPane.showMessageDialog(null, "Invalid Credentials");
42             }
43         }
44         else{
45             if(username.equals("Doctor") & password.equals("123456")){
46                 JOptionPane.showMessageDialog(null, "Success");
47             }
48             else{
49                 JOptionPane.showMessageDialog(null, "Invalid Credentials");
50             }
51         }
52     }
53 }

```

Fig 4.3: Login page code snippet

```

if(r2.isSelected() == true)
{
    username = r1.getText();
    password = r2.getPassword();
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection con1 = DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
        Statement stat1=con1.createStatement();
        String pname="select * from doctor where username='"+username+"' and password='"+password+"'";
        ResultSet rs1=stat1.executeQuery(pname);
        if(rs1.next())
        {
            pid=rs1.getString(1);
            did=rs1.getInt(2);
            //System.out.println(did);
        }
        rs1.close();
        stat1.close();
        con1.close();
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection con2 = DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
        Statement stat2=con2.createStatement();
        String query="select * from doctor where username='"+username+"' and password='"+password+"'";
        ResultSet rs2=stat2.executeQuery(query);
        if(rs2.next())
        {
            Doctor_Login dl=new Doctor_Login(po,username,password,pid,did);
            dl.setVisible(true);
        }
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}

```

Fig 4.4: Login page code snippet 1

```

private void r2ActionPerformed(java.awt.event.ActionEvent evt)
{
    if(r2.isSelected() == true)
    {
        Patient_Login pl=new Patient_Login(po,username,password,pid,did);
        pl.setVisible(true);
        this.setVisible(false);
    }
    else
    {
        JOptionPane.showMessageDialog(null,"invalid credentials");
    }
}
rs.close();
stmt.close();
con.close();
}
catch(Exception e)
{
    System.out.println(e);
}

private void jButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    reset_patient rp=new reset_patient();
    rp.setVisible(true);
    this.setVisible(false);
}

private void jButtonActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    reset_doctor rd=new reset_doctor();
    rd.setVisible(true);
    this.setVisible(false);
}

```

Fig 4.5: Login page code snippet 2

The register patient/doctor frames contains text fields that ask the user for their details that are eventually stored in the database. These details have several text constraints for example, the

contact number cannot be more than 10 digits and the email provided by the user should be of the correct format etc. Here, the username is the primary key which means that if a user enters a username that is already present in the database, that account will not be created. All the details added in the Create Patient frame are added to the Patient table and the details added in the Create Doctor frame are added to the Doctor table.

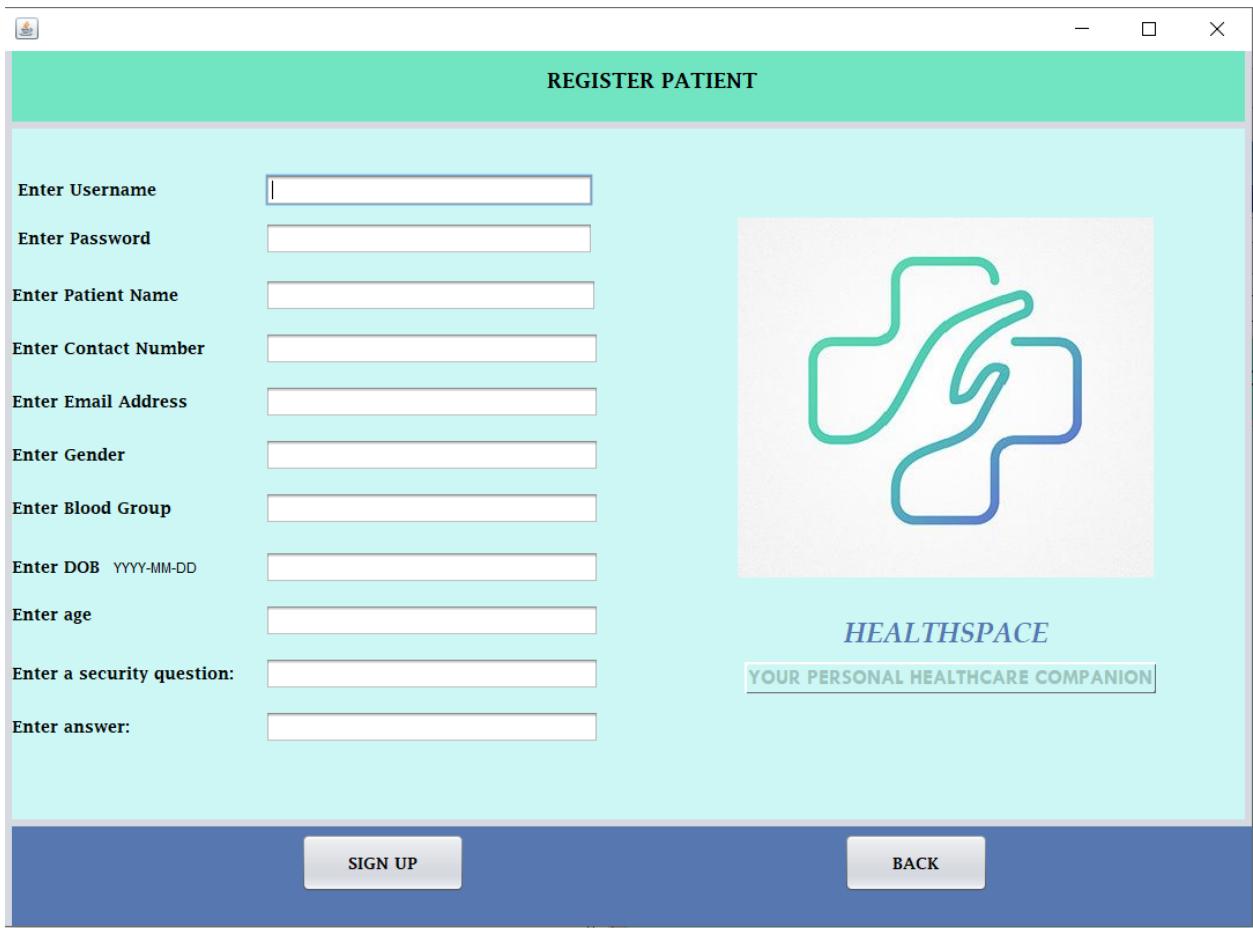


Fig 4.6: Create patient

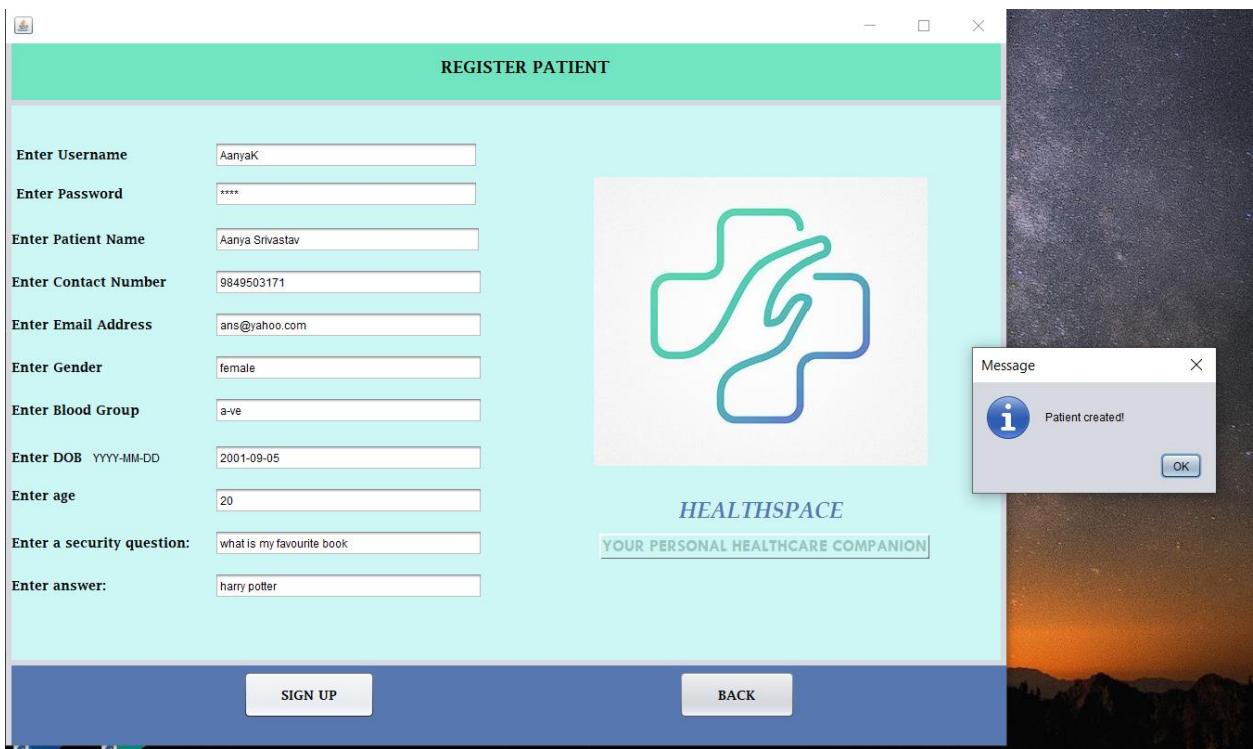


Fig 4.7: Patient created

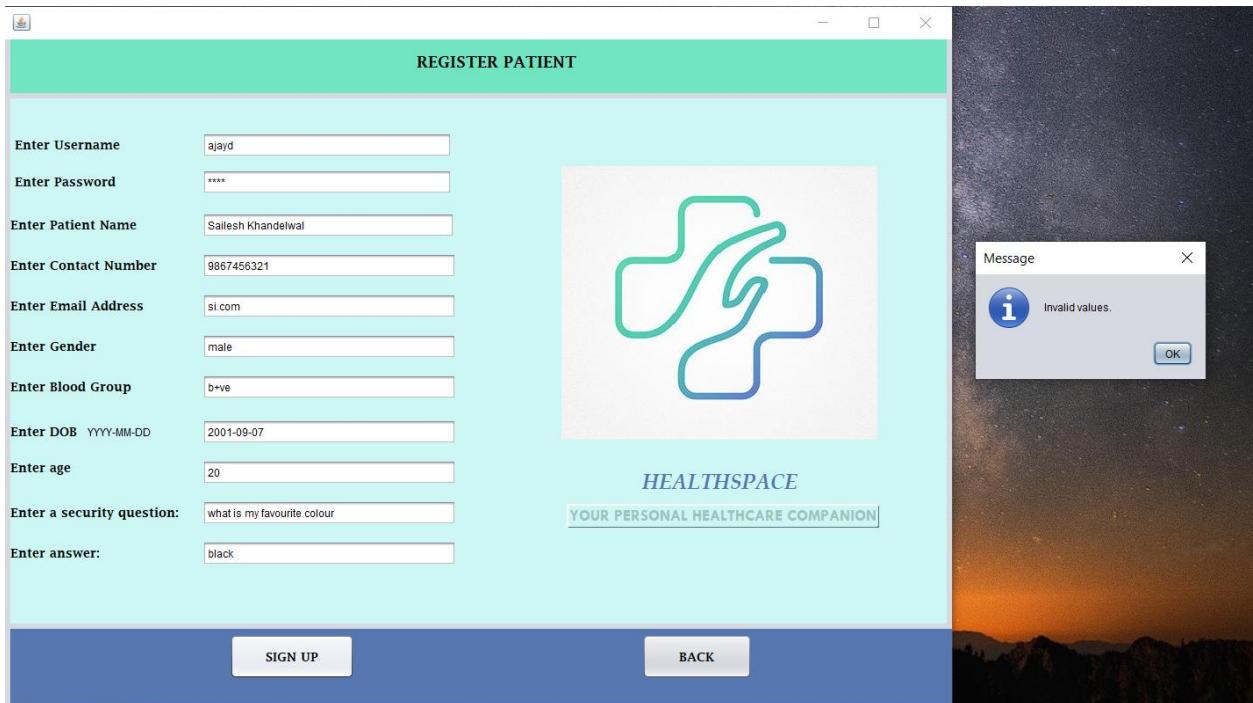


Fig 4.8: Invalid patient registration

Health_space - NetBeans IDE 8.2 RC

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects Files Services Start Page JoptionPane.java

Source Packages Health_space default package

- gui
- Health_space
- Source Packages
- default package
 - CreatePatient.java
 - CreateDoctor.java
 - Doctor_Login.java
 - Home.java
 - Label
 - JoptionPane.java
 - List_app.java
 - Logout.java
 - billing.java
 - book_appointment.java
 - cancel_app.java
 - d_healthstatus.java
 - d_label.java
 - p_details.java
 - p_healthStatus.java
 - p_list.java
 - p_viewAppointments.java
 - reset_doctor.java
 - reset_patient.java
 - show_report.java

Members <empty>

```

private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    String username=jp1.getText();
    String password=jp2.getText();
    String patientname=jp3.getText();
    String contact=jp4.getText();
    String email=jp5.getText();
    String gender=jp6.getText();
    String bgroup=jp7.getText();
    String dob=jp8.getText();
    String question=jp9.getText();
    String answer=jp10.getText();
    String answeren=jp11.getText();
    int age=Integer.parseInt(jp12.getText());
    try {
        if(!(email.contains("@")||!email.contains(".")))
            throw new Exception("Email not valid");
        Class.forName("com.mysql.jdbc.Driver");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta15");
        Statement statm=con.createStatement();
        String query="Insert into patient(Name,Contact,Email,Bgroup,Gender,Username,Password,dob,age,question,answer) values('"+patientname+"','"+contact+"','"+email+"','"+bgroup+"','"+gender+"','"+username+"','"+password+"','"+dob+"','"+age+"','"+question+"','"+answeren+"')";
        statm.executeUpdate(query);
        JOptionPane.showMessageDialog(null,"Patient created!");
        statm.close();
        con.close();
    }
    catch(Exception e)
    {
        System.out.println(e);
        JOptionPane.showMessageDialog(null,"Invalid values.");
    }
}
< private void jButtonMouseClicked(java.awt.event.MouseEvent evt) {>

```

Output - Health_space (run)

Fig 4.9: Create patient code snippet



Fig 4.10: Create doctor

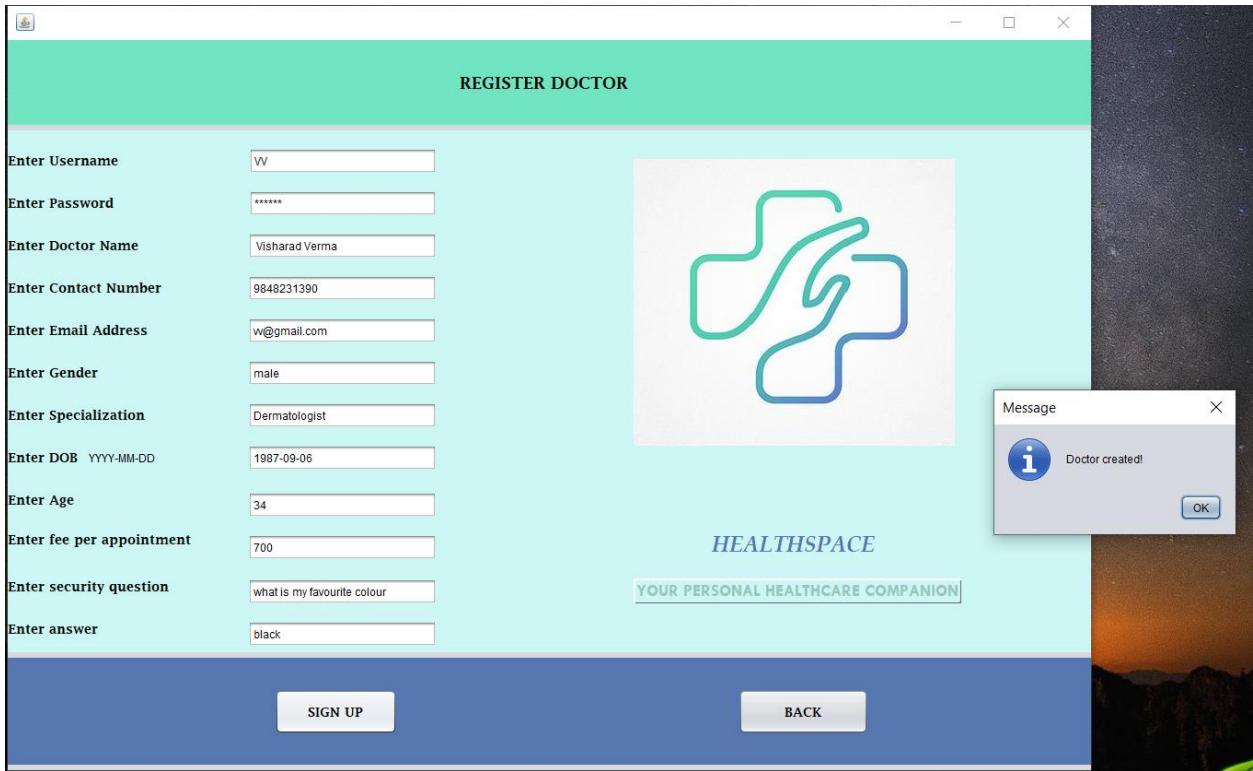


Fig 4.11: Doctor created

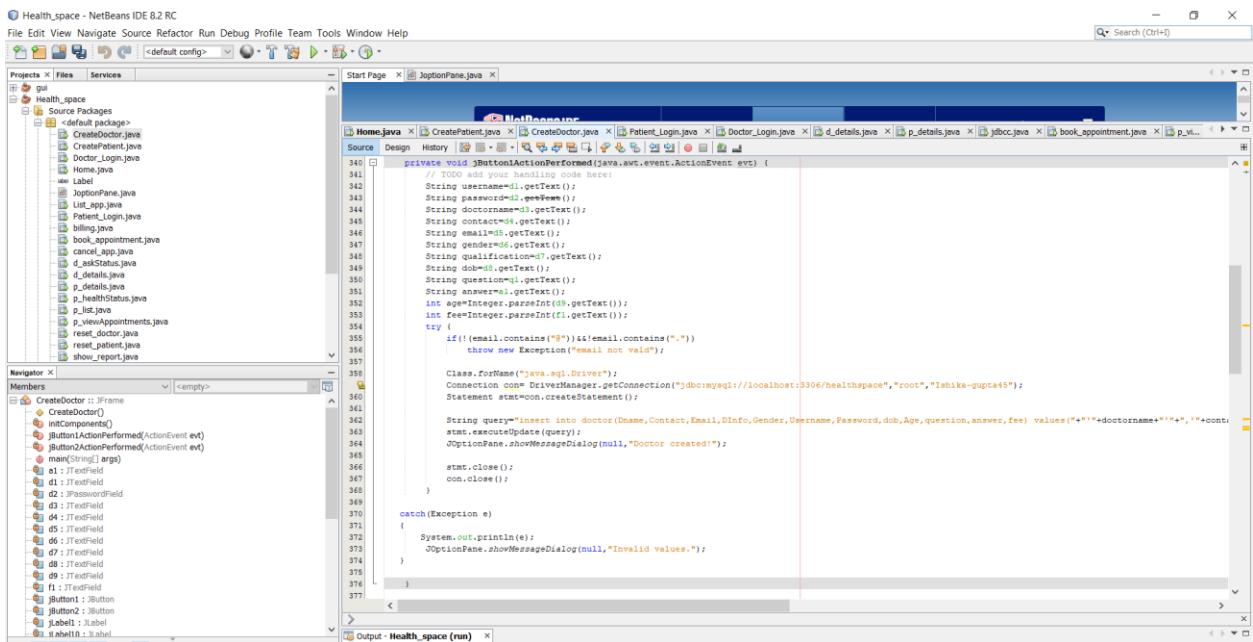


Fig 4.12: Create doctor code snippet

If an existing user forgets his/her password, the ‘Forgot Password’ button is present to assist them. Depending on whether the user is a patient or a doctor, the user must choose the appropriate ‘Forgot Password’ button. This frame asks the user for his/her ID which then accesses the security question the user entered when his/her account was created in the first place. With the security question displayed, the user is required to enter the correct answer and their new password. If the answer matches with the one present in the database, the password update is successful otherwise it is invalidated.

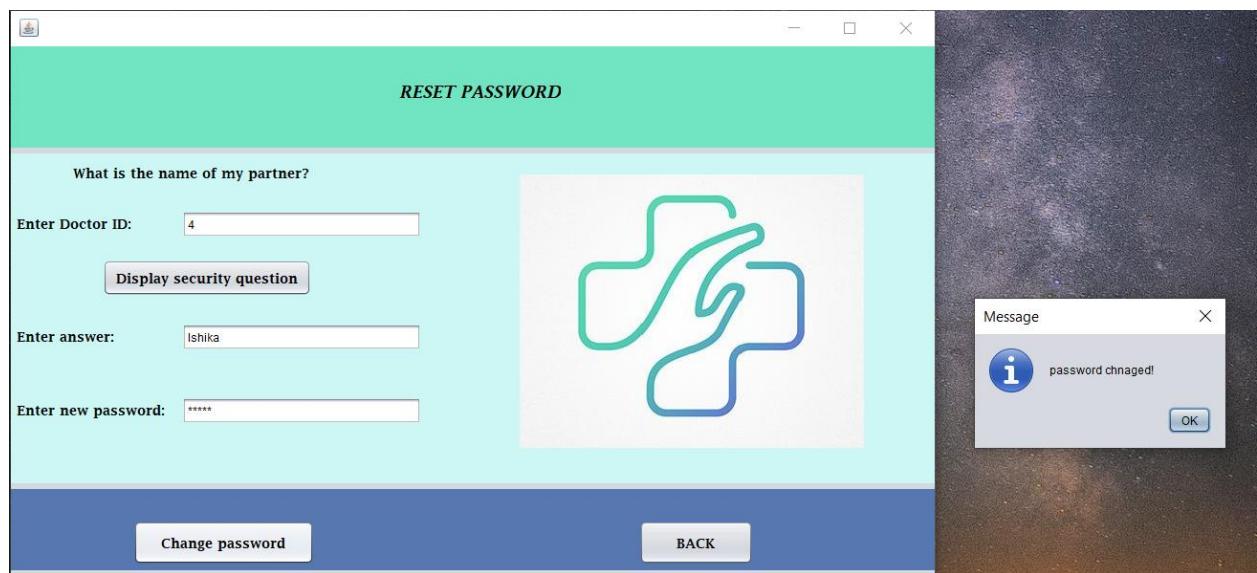


Fig 4.13: Reset doctor password

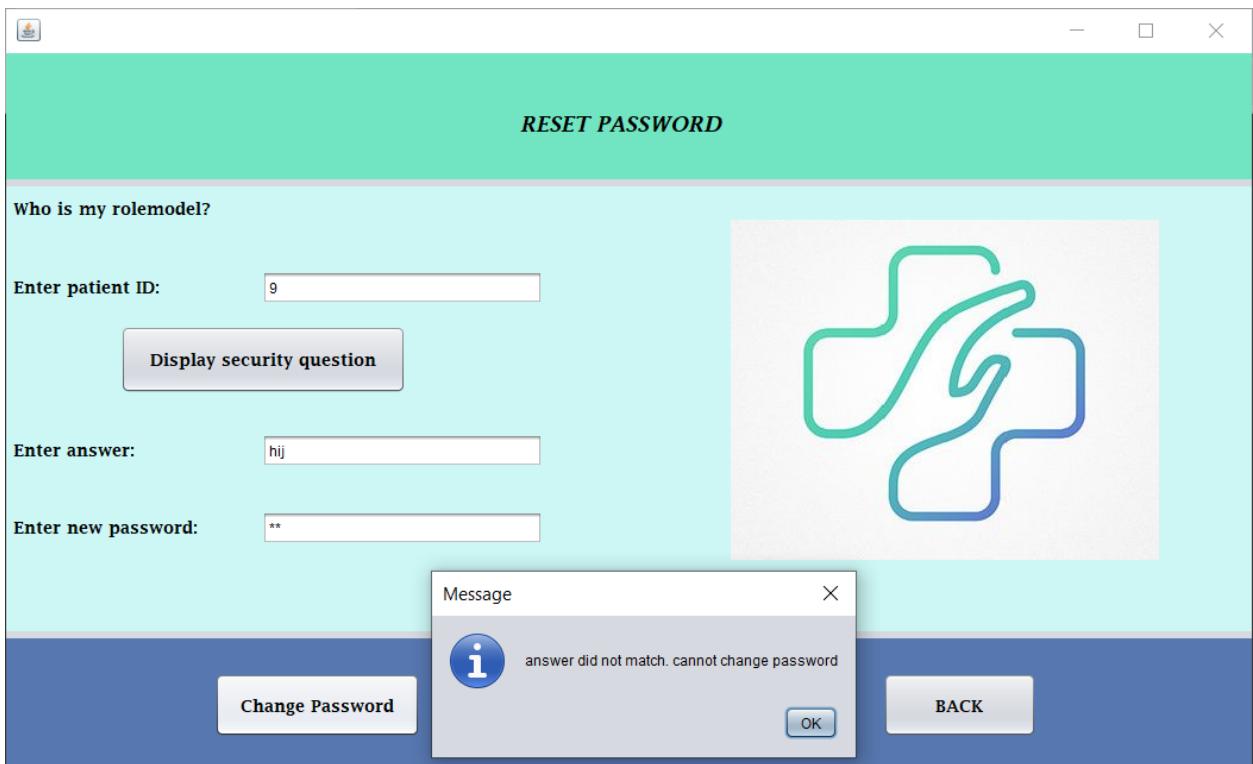


Fig 4.14: Invalid password change

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Health_space - NetBeans IDE 8.2 RC
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard Java development tools.
- Project Explorer:** Shows files like List_app.java, Billing.java, Book_Appointment.java, etc., and a Test Packages section.
- Navigator:** Shows members of the reset_patient class.
- Code Editor:** Displays the following Java code for the JButtonActionPerformed method:

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int p_id=Integer.parseInt(t1.getText());
    String password=t2.getText();
    String ans=t3.getText();
    String answer=ans.toLowerCase();
    String question="";
    String Aname="";
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
        Statement stat=con.createStatement();
        String query="select * from patient where pid='"+p_id+"'";
        ResultSet rs=stat.executeQuery(query);
        while(rs.next())
        {
            ans=rs.getString(12);
        }
        if(!answer.equalsIgnoreCase(ans))
        {
            JOptionPane.showMessageDialog(null,"Answer did not match. cannot change password");
        }
        rs.close();
        stat.close();
        con.close();
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}

```

- Output:** Shows the output for the run configuration.

Fig 4.15: Reset password code snippet 1

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Health_space - NetBeans IDE 8.2 RC
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard Java development tools.
- Project Explorer:** Shows files like List_app.java, Billing.java, Book_Appointment.java, etc., and a Test Packages section.
- Navigator:** Shows members of the reset_patient class.
- Code Editor:** Displays the following Java code for the JButton2ActionPerformed method:

```

try
{
    Class.forName("java.sql.DriverManager");
    Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
    Statement stat=con.createStatement();
    String query="update patient set Password='"+password+"' where pid='"+p_id+"' and answer='"+answer+"'";
    stat.executeUpdate(query);
    if(answer.equalsIgnoreCase(ans))
    {
        JOptionPane.showMessageDialog(null,"password changed");
    }
    stat.close();
    con.close();
}
catch(Exception e)
{
    System.out.println(e);
}

```

Below this, the code for the JButton3ActionPerformed method is partially visible:

```

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int p_id=Integer.parseInt(t1.getText());
    String password=t2.getText();
    String ans=t3.getText();
    String answer=ans.toLowerCase();
    String question="";
    String Aname="";
    try
    {

```

- Output:** Shows the output for the run configuration.

Fig 4.16: Reset password code snippet 2

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Health_space - NetBeans IDE 8.2 RC
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations, search, and project navigation.
- Project Explorer:** Shows a Java project named "Health_space" with packages like "com.health_space" containing classes such as List_app, Patient_Login, billing.java, book_appointment.java, cancel_app.java, d_details.java, d_details.java, p_details.java, p_healthStatus.java, p_list.java, P_view_appointments.java, reset_doctor.java, reset_patient.java, view_patient.java, view_report.java, view_doctors.java, view_prescription.java, write_prescription.java, and write_report.java. It also lists "images", "Test Packages", and "Libraries" including MySQL Connector Java 8.0.23.jar and JUnit 4.12 (Default).
- Code Editor:** The main window displays Java code for a class named "JOptionPane.java". The code handles button actions, connects to a MySQL database, and performs various database queries related to patient records and prescription status.
- Output:** Shows the output for the "Health_space (run)" configuration.

The Patient Login page contains the several features a patient has access to. These features are, 'View My Details', 'View Your Appointments', 'View Doctors', 'Book an Appointment', 'Cancel your appointment', 'Update health status', 'View Report' and 'View Your Prescription'.

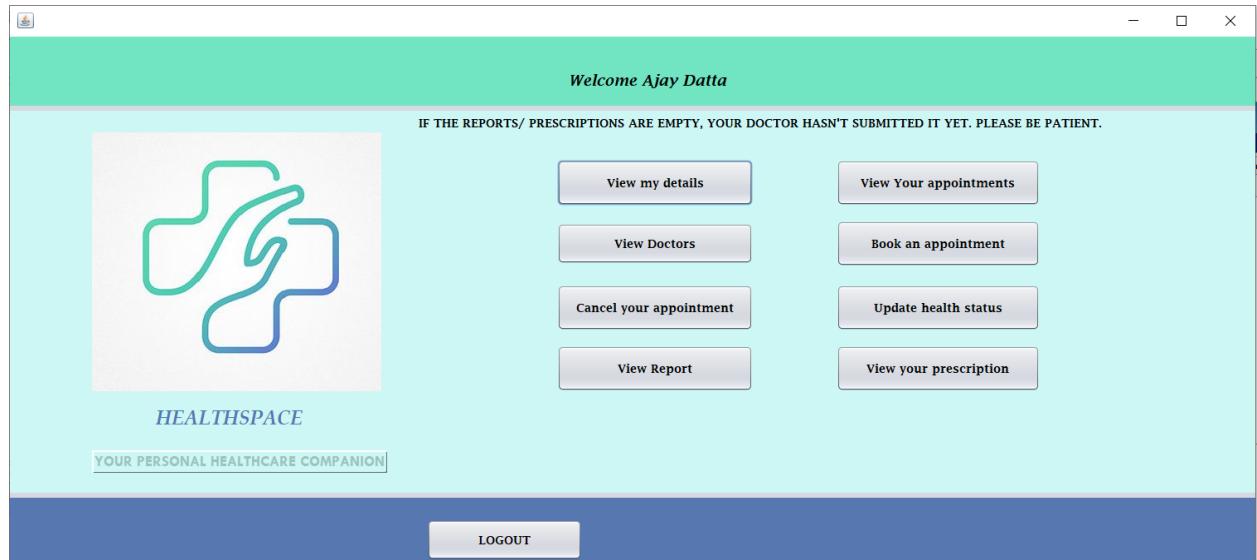


Fig 4.17: Patient login features

```

private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Home h=new Home();
    h.setVisible(true);
    this.setVisible(false);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here
    p_details pd=new p_details(u,p,pid,did);
    pd.setVisible(true);
    pd.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(false);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    book_appointment ba=new book_appointment(n,u,p,pid,did);
    ba.setVisible(true);
    this.setVisible(false);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    p_viewAppointments pva=new p_viewAppointments(n,u,p,pid,did);
    pva.setVisible(true);
    pva.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(false);
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    cancel_app ca=new cancel_app(u,p,n,pid,did);
    ca.setVisible(true);
    ca.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.setVisible(false);
}

```

Fig 4.18: Patient login code snippet

The Patient Login page contains the several features a patient has access to. These features are, 'View My Details', 'List of patients', 'List of appointments', 'Write Report' and 'Write Prescription'. The logout button logs the user out of their account.

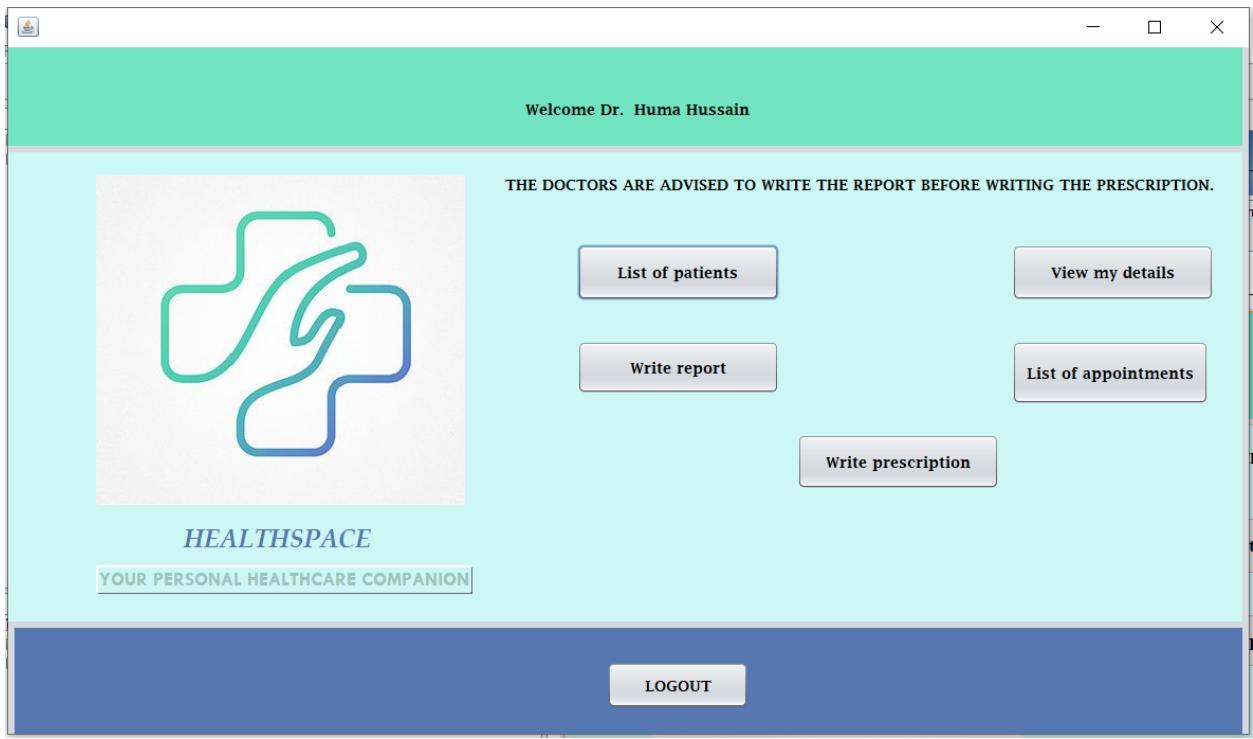


Fig 4.19: Doctor login features

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    List_app la=new List_app(u,p,n,pid,did);
    la.setVisible(true);
    this.setVisible(false);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    d_details dd=new d_details(u,p,n,pid,did);
    dd.setVisible(true);
    this.setVisible(false);
}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    p_list pl=new p_list(u,p,n,pid,did);
    pl.setVisible(true);
    this.setVisible(false);
}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    write_prescription wp=new write_prescription(u,p,n,pid,did);
    wp.setVisible(true);
    this.setVisible(false);
}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    write_report wr=new write_report(u,p,n,pid,did);
    wr.setVisible(true);
    this.setVisible(false);
}

```

Fig 4.20: Doctor login code snippet

The Patient/Doctor details page displays all the information about the user. This frame comes with the ‘Update Information’ feature where if the user wishes to change or update any of the information about themselves, they can make changes in the respective text fields and click on ‘Update Information’ button. The changes will be reflected in the database.

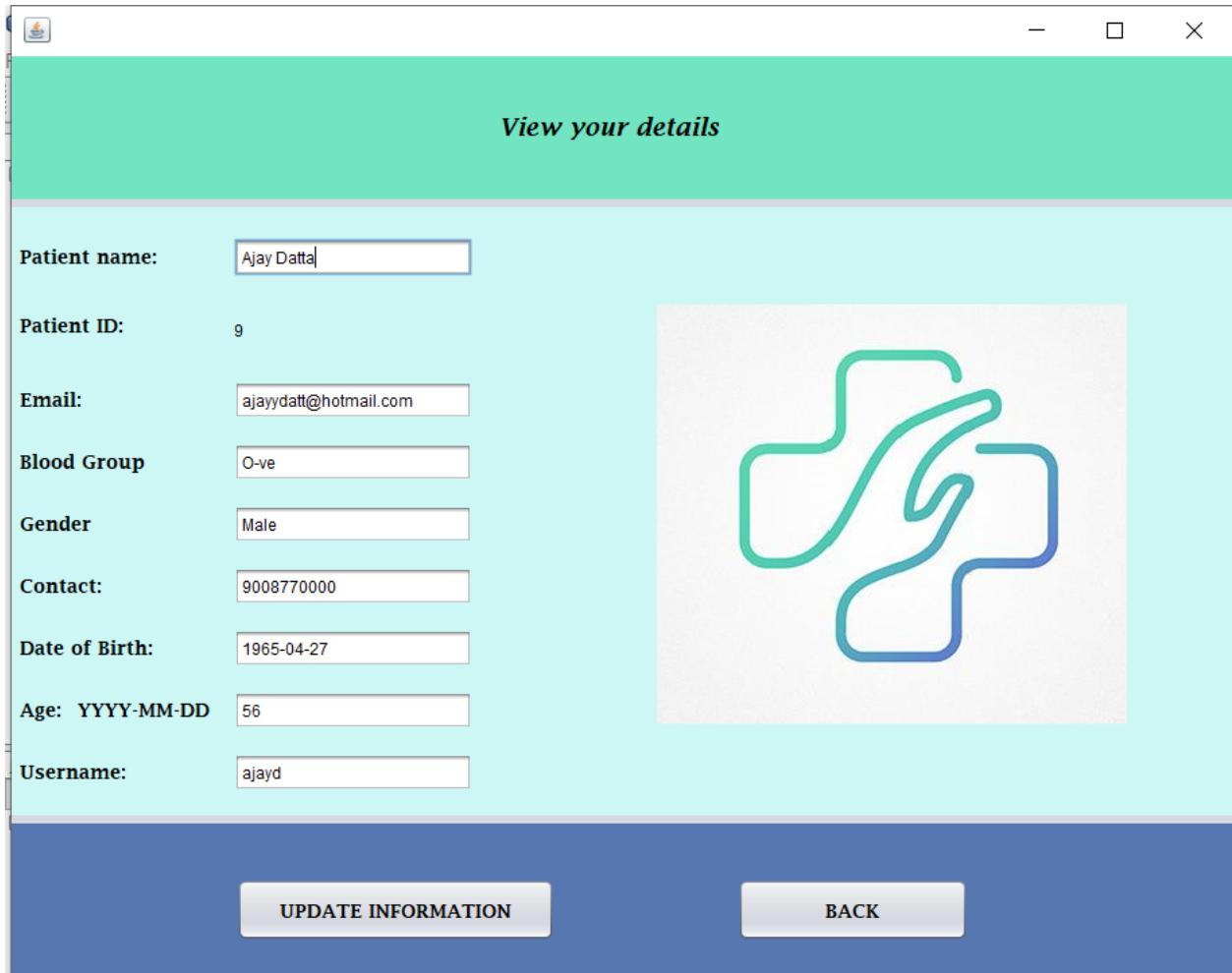


Fig 4.21: Patient details

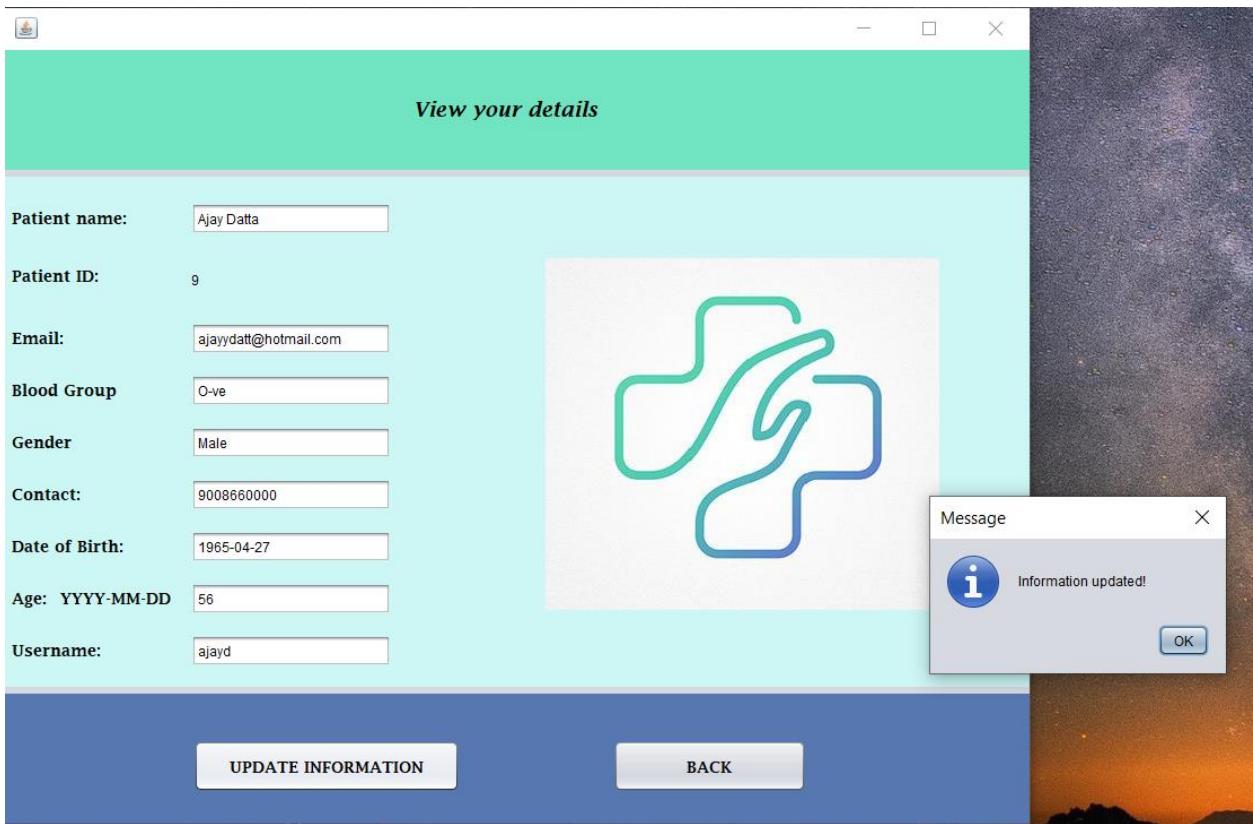


Fig 4.22: Updating patient details

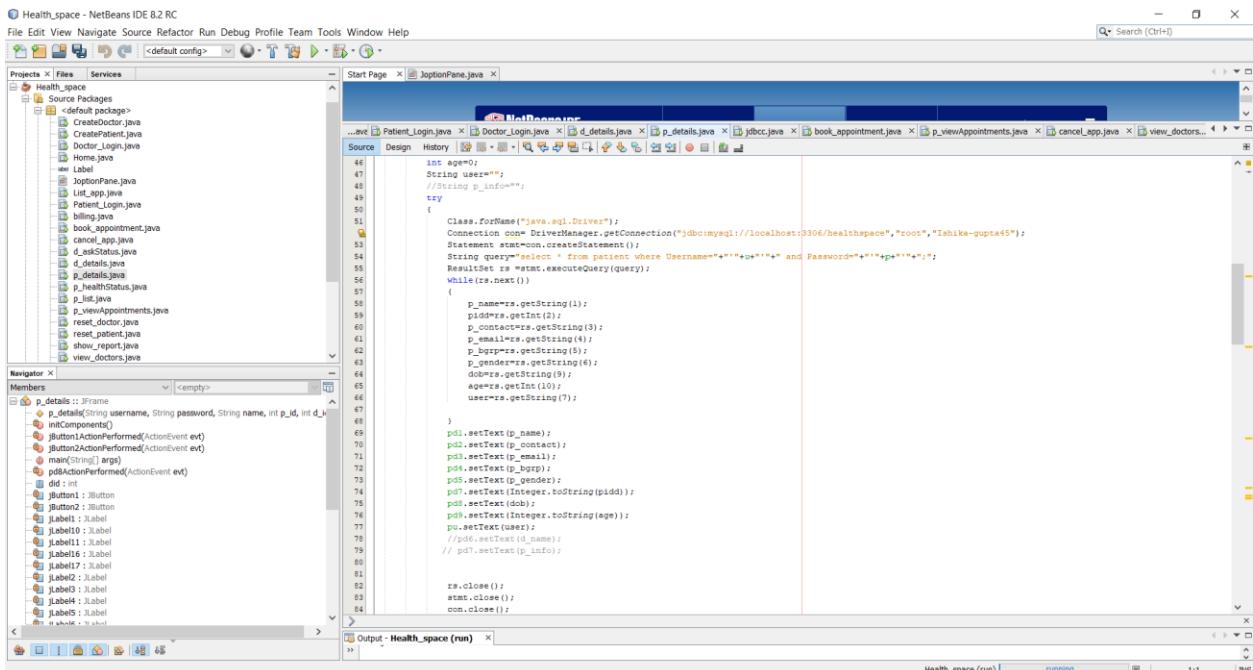


Fig 4.23: Patient details code snippet

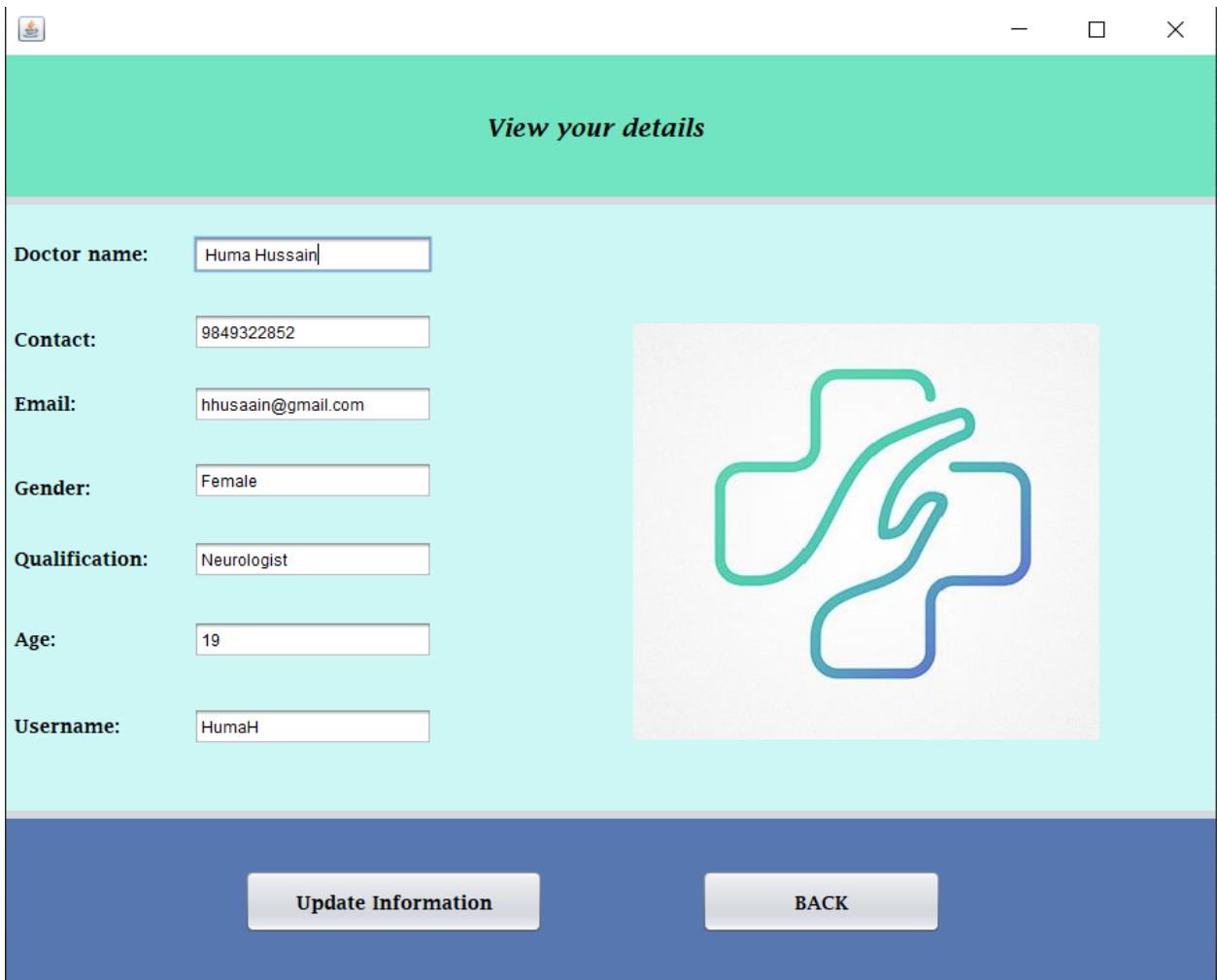


Fig 4.24: Doctor details

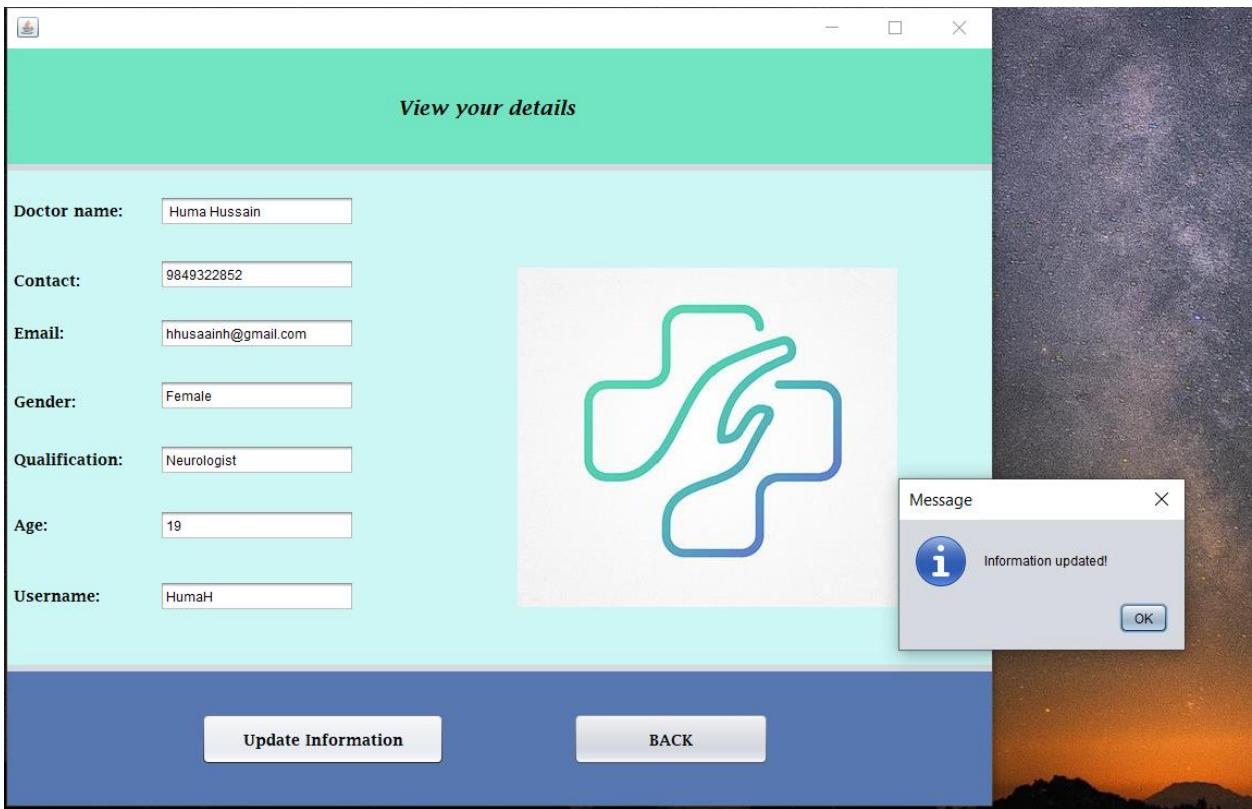
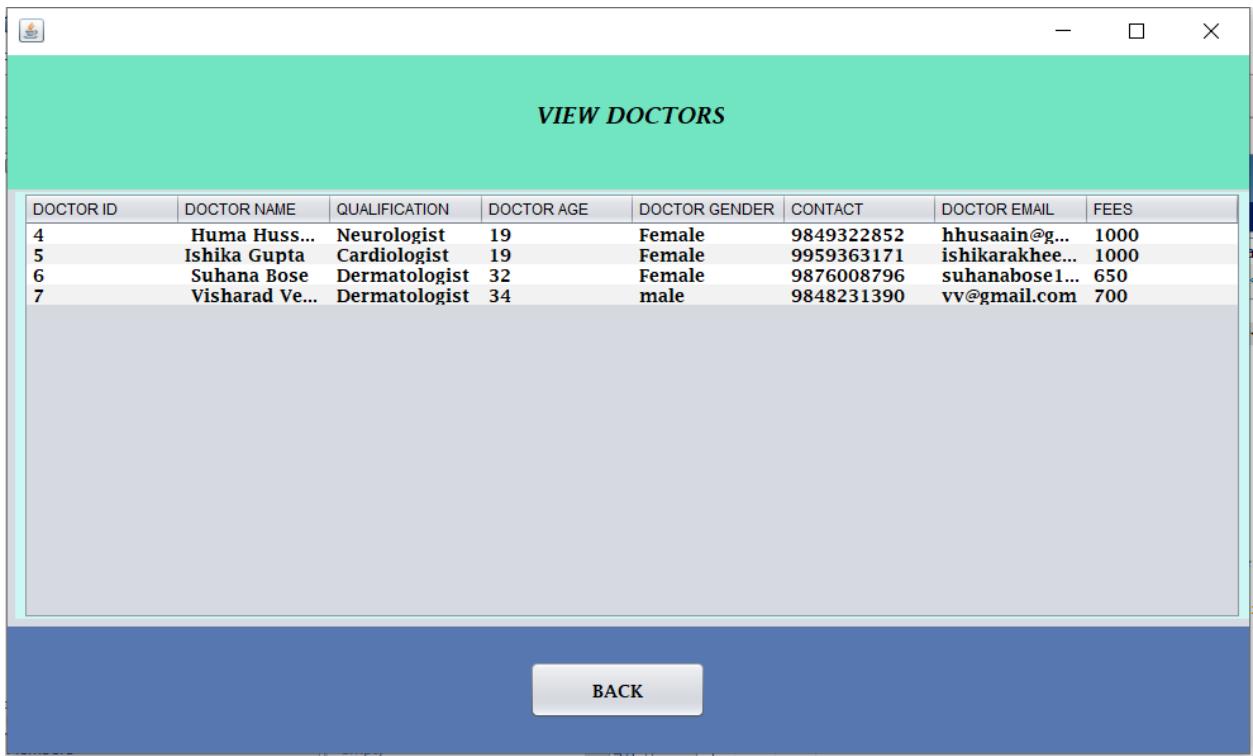


Fig 4.25: Doctor details update

Fig 4.26: Doctor details code snippet

The ‘View Doctors’ feature in the patient login portal displays all the details of all the doctors present in the database for the patient to view. With this feature, the patient can view the qualifications of the doctors and book an appointment with a doctor of their liking.



The screenshot shows a Windows application window titled "VIEW DOCTORS". The main content area displays a table of doctor information:

DOCTOR ID	DOCTOR NAME	QUALIFICATION	DOCTOR AGE	DOCTOR GENDER	CONTACT	DOCTOR EMAIL	FEES
4	Huma Huss...	Neurologist	19	Female	9849322852	hhusain@g...	1000
5	Ishika Gupta	Cardiologist	19	Female	9959363171	ishikarakhee...	1000
6	Suhana Bose	Dermatologist	32	Female	9876008796	suhananbose1...	650
7	Visharad Ve...	Dermatologist	34	male	9848231390	vv@gmail.com	700

At the bottom of the window is a blue footer bar with a "BACK" button.

Fig 4.27: View doctors

```

public void view_doctors(String username, String password, String name, int p_id, int d_id)
{
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
        Statement statm=con.createStatement();
        String query="select * from doctor";
        ResultSet res=statm.executeQuery(query);
        while(res.next())
        {
            DefaultTableModel model=(DefaultTableModel) table.getModel();
            String d_name=res.getString(1);
            String d_email=res.getString(2);
            String doctor_contact=res.getString(3);
            String d_qual=res.getString(4);
            String d_qualr=res.getString(5);
            String d_gender=res.getString(6);
            int d_age=res.getInt(10);
            int fees=r.getInt(13);

            model.addRow(new Object[]{doctor_id, d_name, d_qual, d_age, d_gender, doctor_contact, d_email, fees});
        }
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}

```

Fig 4.28: View doctors code snippet

The ‘List of Patients’ feature enables the doctor to view all the patients present in the database. It contains two tables; the first table contains basic information about the patient like their name and patient ID. If the doctor wishes to know further about the patient, they can insert their patient ID in the given text field and all the details about the patient are displayed in the second table. This frame makes use of the information provided in the Patient table.

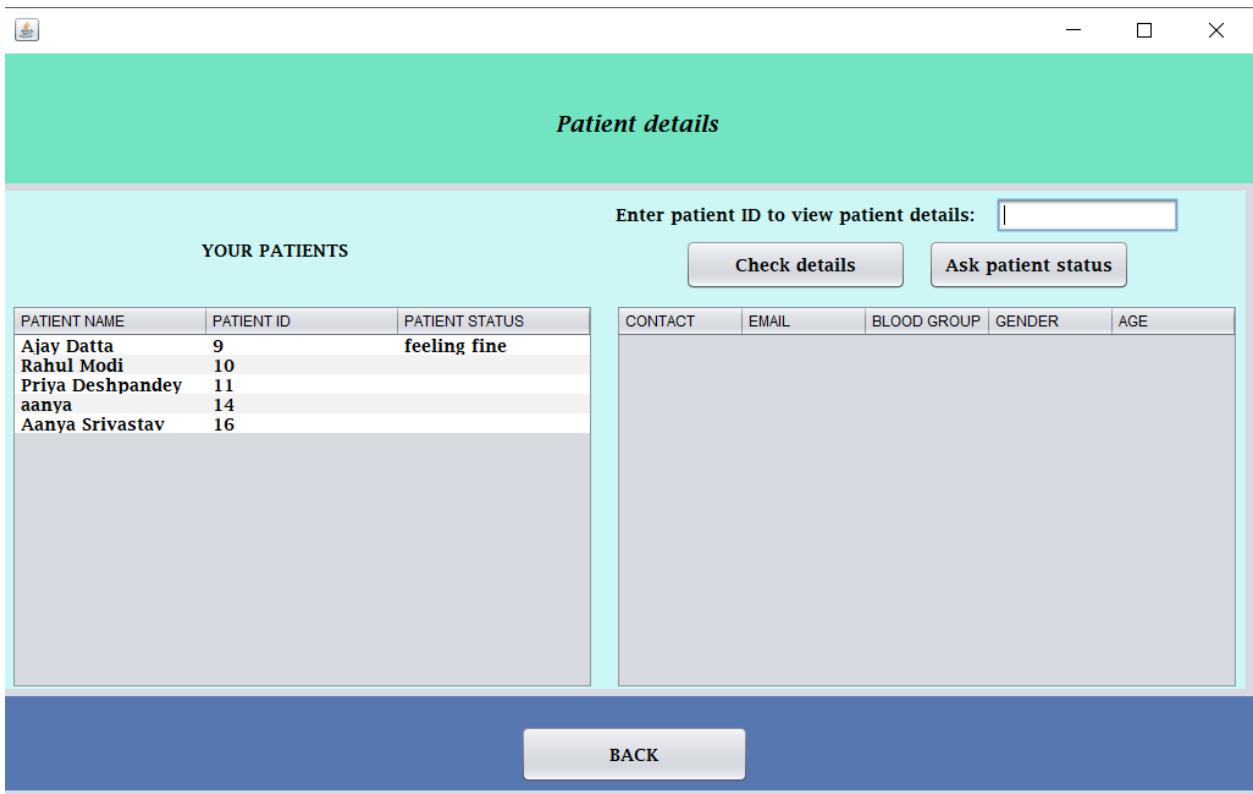


Fig 4.29: List of patients

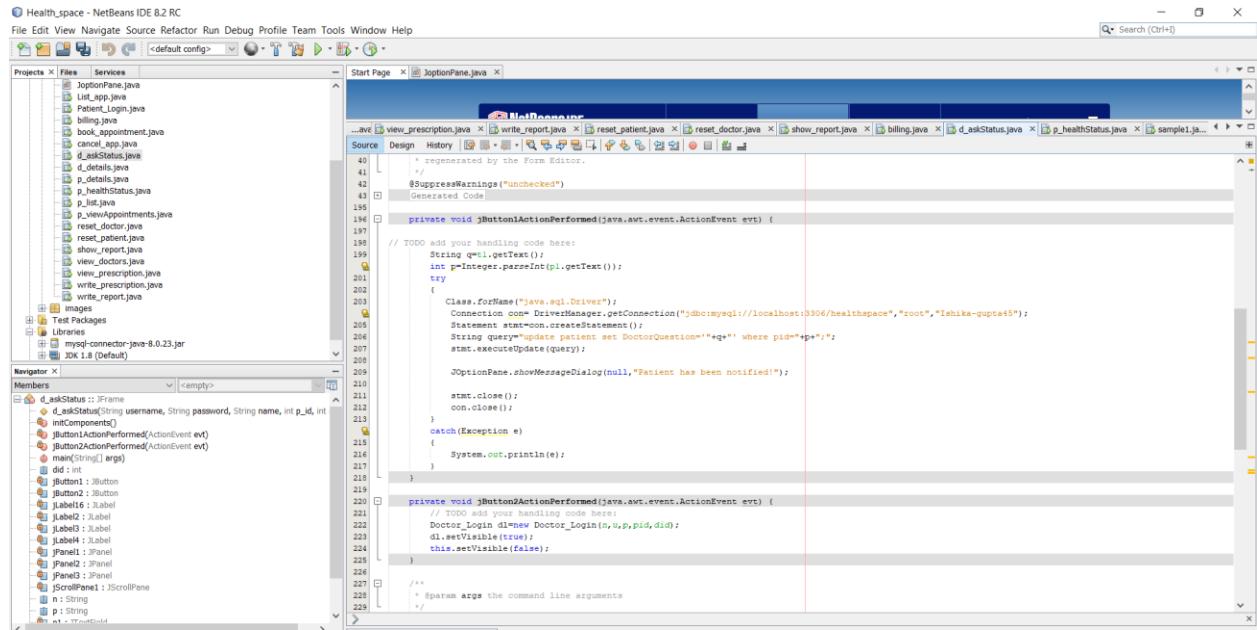
```

public class List_app extends javax.swing.JFrame {
    /**
     * Creates new form List_app
     */
    public List_app(String username, String password, String name, int pid, int d_id) {
        initComponents();
        setUsername();
        setPassword();
        mName=name;
        pid=pid;
        d_id=d_id;
        try {
            Class.forName("java.sql.Driver");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
            Statement stat=con.createStatement();
            String query="select * from appointment where did='"+d_id+"'";
            ResultSet rs=stat.executeQuery(query);
            while(rs.next())
            {
                DefaultTableModel model=(DefaultTableModel) table.getModel();
                String p_name=rs.getString(1);
                String time=rs.getString(3);
                String date=rs.getString(4);
                int p_id=rs.getInt(5);
                int a_id=rs.getInt(7);
                String p_info=rs.getString(8);
                model.addRow(new Object[]{ p_id, p_name, p_info, time, date});
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Fig 4.30: List of patients code snippet

The ‘Ask Patient Status’ button in the ‘List of Patients’ frame enables the doctor to have a one-on-one, efficient conversation with the patient about their wellbeing. This input is added to the Appointment table.



```

    private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        String q1=txtText.getText();
        int p=Integer.parseInt(q1.getText());
        try
        {
            Class.forName("java.sql.DriverManager");
            Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta5");
            Statement stat=con.createStatement();
            String query="update patient set DoctorQuestion='"+q1+"' where pid='"+p+"'";
            stat.executeUpdate(query);
            JOptionPane.showMessageDialog(null,"Patient has been notified!");
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }

    private void jButtonActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        Doctor_Login d=new Doctor_Login(n,u,p,pid,did);
        d.setVisible(true);
        this.setVisible(false);
    }

    /**
     * @param args the command line arguments
     */
}

```

Fig 4.31: Ask patient code snippet

The ‘Update Health Status’ feature in the patient login portal creates a direct connection between the patient and the doctor. This feature is analogous to a chat feature between the patient and a doctor. A question that has been asked by a doctor to a patient using the ‘Ask Patient Status’ feature is projected in the patient’s portal in this frame. The patient can reply whatever’s relevant to the question and the answer will be reflected in the first table in the ‘List of Patients’ frame in the doctors portal. This feature is prime in creating a more personal bond between the doctor and the patient.

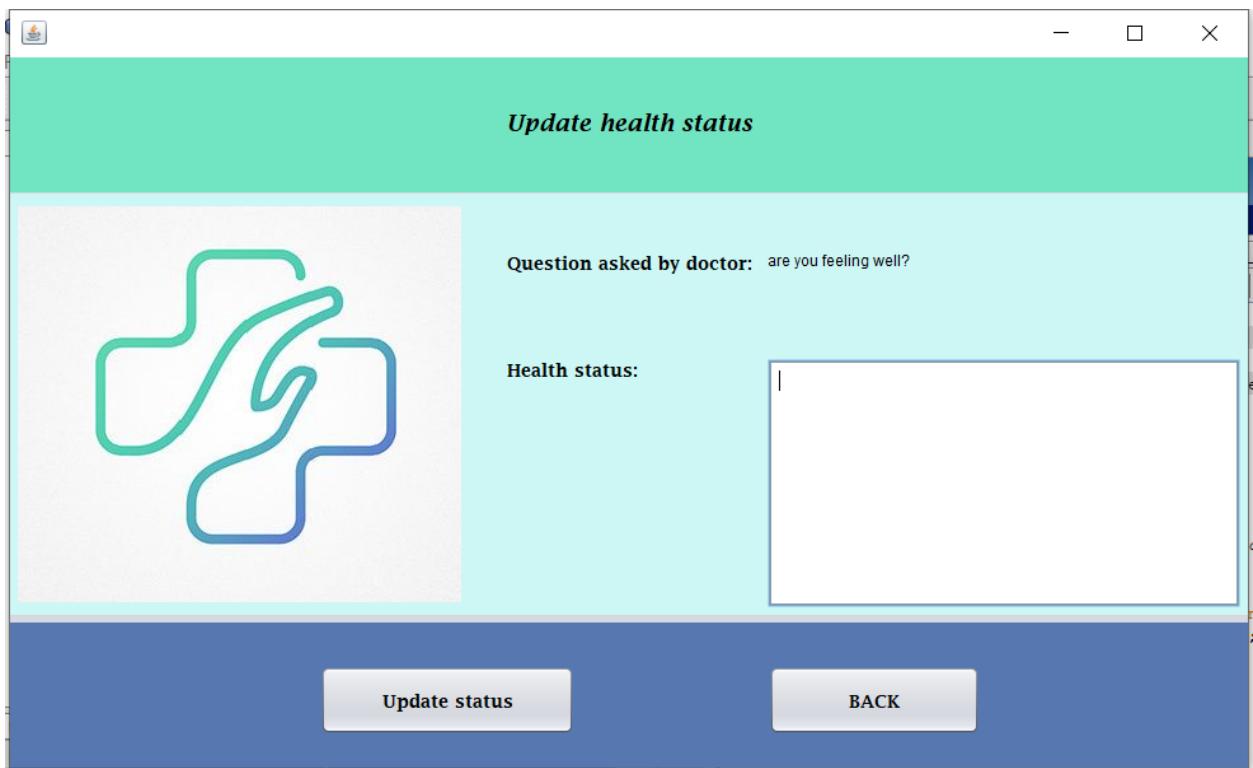


Fig 4.32: Update health status

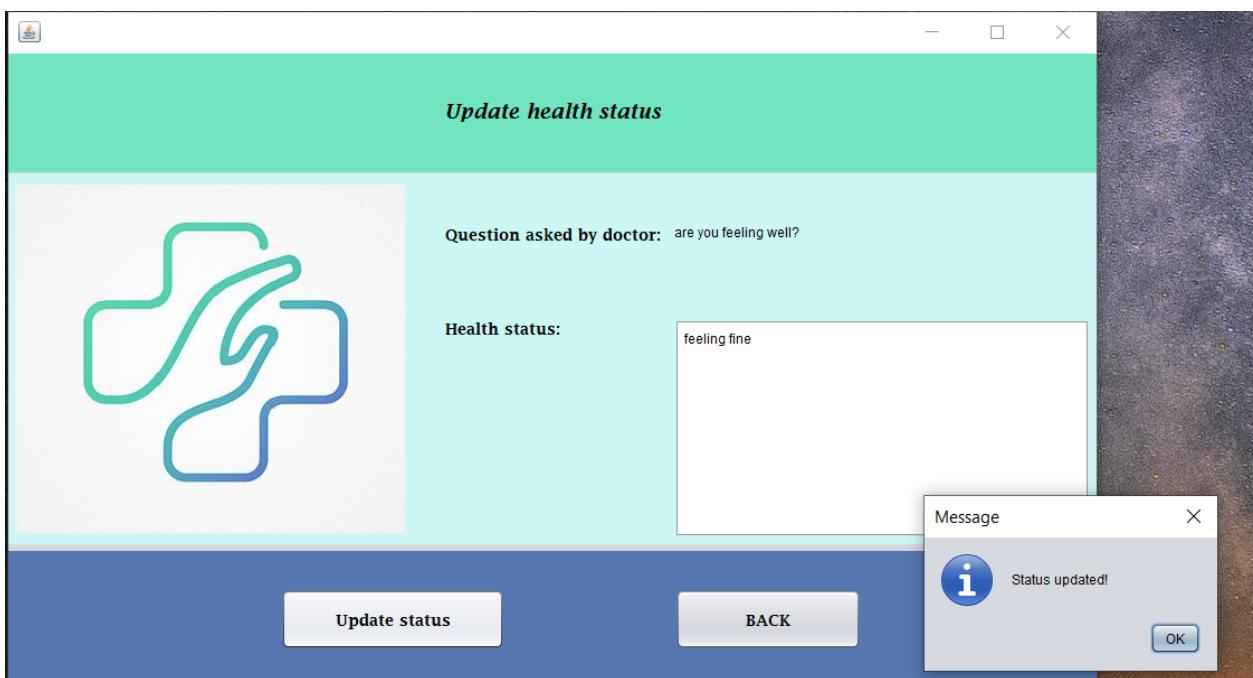


Fig 4.33: Updating health status

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Health_space - NetBeans IDE 8.2 RC
- Toolbar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Project Explorer:** Shows files like Home.java, JoptionPane.java, List_app.java, Patient_Login.java, Billing.java, book_appointment.java, cancel_app.java, d_healthStatus.java, d_details.java, p_healthStatus.java, p_details.java, p_healthStatus.java, p_list.java, p_viewAppointments.java, reset_doctor.java, reset_patient.java, show_report.java, view_doctors.java, view_prescription.java, write_prescription.java, write_report.java.
- Code Editor:** Displays the `p_healthStatus` class with its constructor and a method for updating patient status. The code uses JDBC to connect to a MySQL database and execute an update query.
- Output:** Shows the output for the Health_space run.

```

20     */
21     // Creates new form p_healthStatus
22     /*
23     public static String u;
24     public static String p;
25     public static String n;
26     public static int pid;
27     public static int did;
28
29
30     public p_healthStatus(String username, String password, String name, int p_id, int d_id) {
31         initComponents();
32         setUsername(username);
33         setPassword(password);
34         setname(name);
35         pid=p_id;
36         did=d_id;
37         String q="";
38         try
39         {
40             Class.forName("java.sql.DriverManager");
41             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
42             Statement stat=con.createStatement();
43             String query="select * from patient where pid='"+pid+"'";
44             ResultSet rs=stat.executeQuery(query);
45             while(rs.next())
46             {
47                 q=rs.getString(13);
48             }
49             l1.setText(q);
50             rs.close();
51             stat.close();
52             con.close();
53         }
54         catch(Exception e)
55         {
56             System.out.println(e);
57         }
58     }
59
60
61
62
63
64     @SuppressWarnings("unchecked")
65     Generated Code
66
67
68     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
69         // TODO add your handling code here:
70         String xlabel.getText();
71         try
72         {
73             Class.forName("java.sql.DriverManager");
74             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
75             Statement stat=con.createStatement();
76             String query="update patient set HealthStatus='"+q+"' where pid='"+pid+"'";
77             stat.executeUpdate(query);
78             JOptionPane.showMessageDialog(null,"Status updated!");
79             stat.close();
80             con.close();
81         }
82         catch(Exception e)
83         {
84             System.out.println(e);
85         }
86     }
87
88     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
89         // TODO add your handling code here:
90         Patient_Login pl=new Patient_Login(n,s,p,pid,did);
91         pl.setVisible(true);
92         this.setVisible(false);
93     }
94
95     /**
96      * @param args the command line arguments
97     */
98     public static void main(String args[]) {
99         /* Set the Nimbus look and feel */
100        LookAndFeel.setLookAndFeel("Nimbus");
101        //
```

Fig 4.34: Update health status code snippet 1

The screenshot shows the NetBeans IDE interface with the following details:

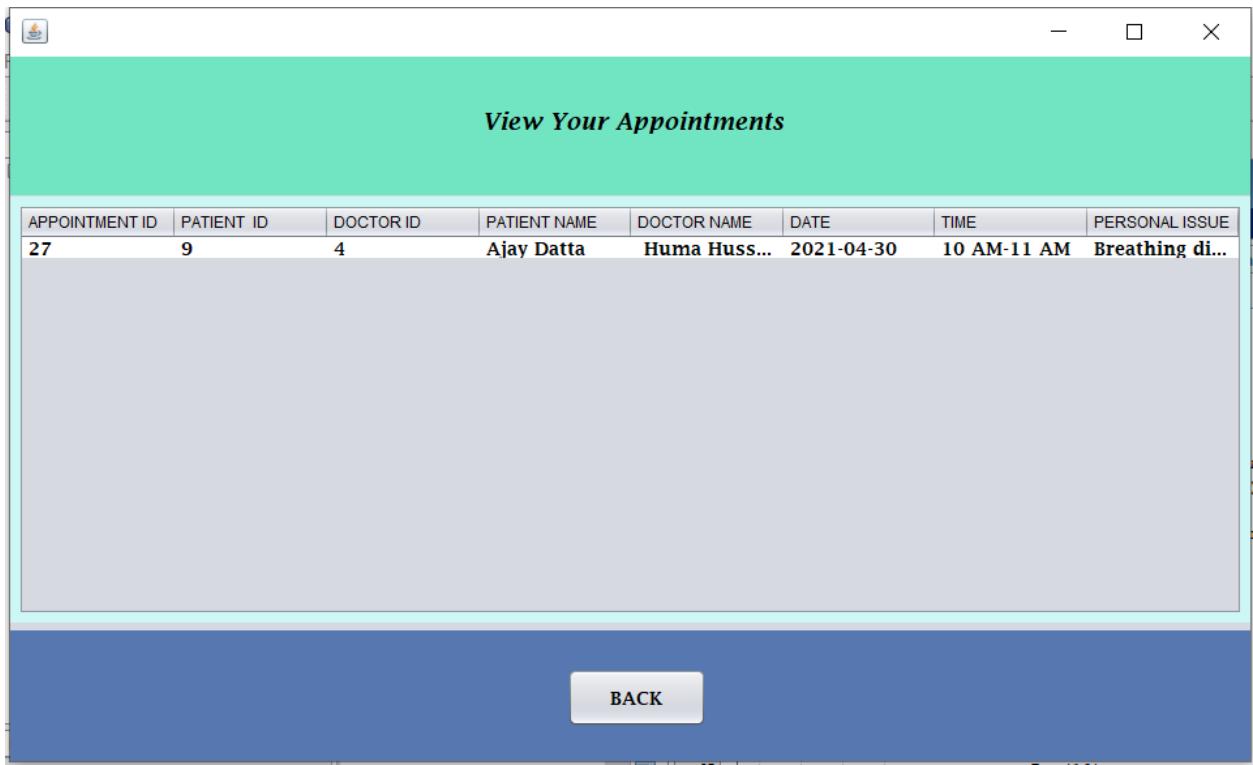
- Title Bar:** Health_space - NetBeans IDE 8.2 RC
- Toolbar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Project Explorer:** Shows files like Home.java, JoptionPane.java, List_app.java, Patient_Login.java, Billing.java, book_appointment.java, cancel_app.java, d_healthStatus.java, d_details.java, p_healthStatus.java, p_details.java, p_healthStatus.java, p_list.java, p_viewAppointments.java, reset_doctor.java, reset_patient.java, show_report.java, view_doctors.java, view_prescription.java, write_prescription.java, write_report.java.
- Code Editor:** Displays the `p_healthStatus` class with its constructor and two methods for updating patient status. The code uses JDBC to connect to a MySQL database and execute update queries.
- Output:** Shows the output for the Health_space run.

```

20     */
21     // Creates new form p_healthStatus
22     /*
23     public static String u;
24     public static String p;
25     public static String n;
26     public static int pid;
27     public static int did;
28
29
30     public p_healthStatus(String username, String password, String name, int p_id, int d_id) {
31         initComponents();
32         setUsername(username);
33         setPassword(password);
34         setname(name);
35         pid=p_id;
36         did=d_id;
37         String xlabel.getText();
38         try
39         {
40             Class.forName("java.sql.DriverManager");
41             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
42             Statement stat=con.createStatement();
43             String query="update patient set HealthStatus='"+q+"' where pid='"+pid+"'";
44             stat.executeUpdate(query);
45             JOptionPane.showMessageDialog(null,"Status updated!");
46             stat.close();
47             con.close();
48         }
49         catch(Exception e)
50         {
51             System.out.println(e);
52         }
53     }
54
55
56
57
58
59     private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
60         // TODO add your handling code here:
61         String xlabel.getText();
62         try
63         {
64             Class.forName("java.sql.DriverManager");
65             Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
66             Statement stat=con.createStatement();
67             String query="update patient set HealthStatus='"+q+"' where pid='"+pid+"'";
68             stat.executeUpdate(query);
69             JOptionPane.showMessageDialog(null,"Status updated!");
70             stat.close();
71             con.close();
72         }
73         catch(Exception e)
74         {
75             System.out.println(e);
76         }
77     }
78
79     private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
80         // TODO add your handling code here:
81         Patient_Login pl=new Patient_Login(n,s,p,pid,did);
82         pl.setVisible(true);
83         this.setVisible(false);
84     }
85
86     /**
87      * @param args the command line arguments
88     */
89     public static void main(String args[]) {
90         /* Set the Nimbus look and feel */
91         LookAndFeel.setLookAndFeel("Nimbus");
92         //
```

Fig 4.35: Update health status code snippet 2

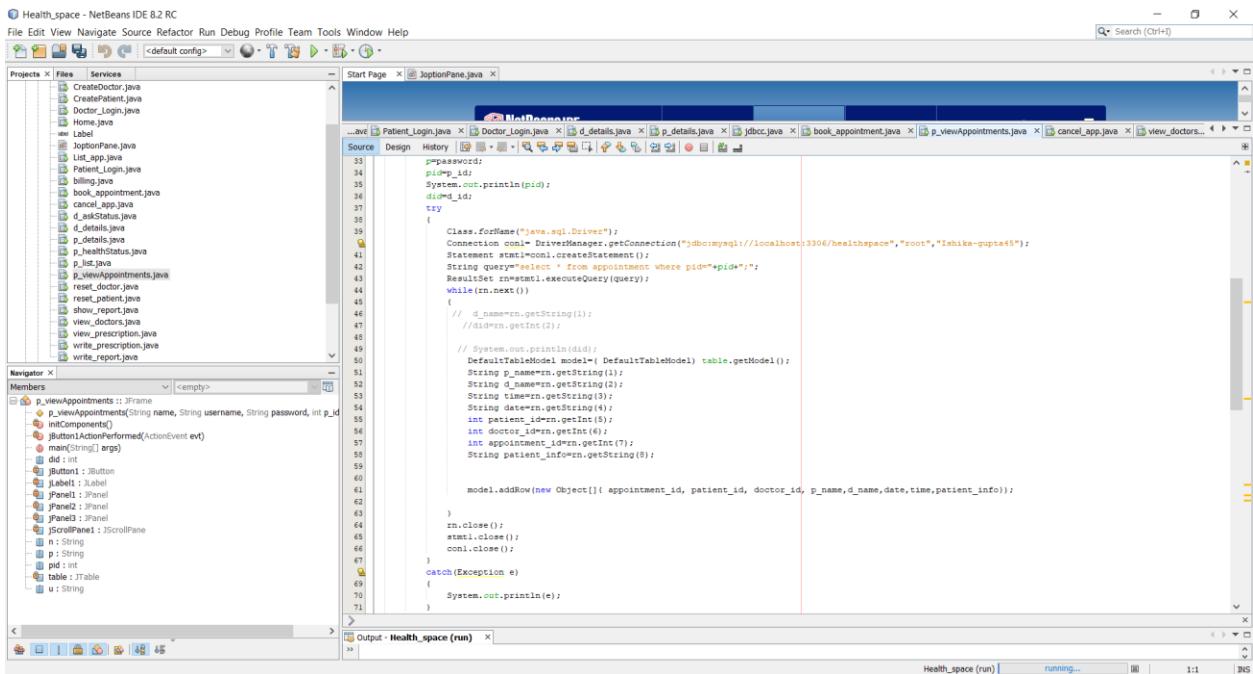
The ‘View Your Appointments’ page in the patient login portal displays a table containing all the information like appointment ID about a patient’s future appointments that have been booked. All of this information is stored in the Appointment table.



A screenshot of a Windows application window titled "View Your Appointments". The window contains a table with one row of data. The columns are labeled: APPOINTMENT ID, PATIENT ID, DOCTOR ID, PATIENT NAME, DOCTOR NAME, DATE, TIME, and PERSONAL ISSUE. The data in the first row is: 27, 9, 4, Ajay Datta, Huma Huss..., 2021-04-30, 10 AM-11 AM, Breathing di... . A "BACK" button is visible at the bottom of the window.

APPOINTMENT ID	PATIENT ID	DOCTOR ID	PATIENT NAME	DOCTOR NAME	DATE	TIME	PERSONAL ISSUE
27	9	4	Ajay Datta	Huma Huss...	2021-04-30	10 AM-11 AM	Breathing di...

Fig 4.36: Patient’s appointments



```

33     String p_name;
34     int pid;
35     System.out.println(pid);
36     doctor_id;
37     try
38     {
39         Class.forName("java.sql.DriverManager");
40         Connection con= DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
41         Statement stat1=con.createStatement();
42         String query="select * from appointment Where pid='"+pid+"'";
43         ResultSet rn=stat1.executeQuery(query);
44         while(rn.next())
45         {
46             // d_name=rn.getString(1);
47             // did=rn.getInt(2);
48
49             // System.out.println(did);
50             DefaultTableModel model=(DefaultTableModel) table.getModel();
51             String p_name=rn.getString(1);
52             String d_name=rn.getString(2);
53             String time=rn.getString(3);
54             String date=rn.getString(4);
55             int patient_id=rn.getInt(5);
56             int doctor_id=rn.getInt(6);
57             int appointment_id=rn.getInt(7);
58             String patient_info=rn.getString(8);
59
60             model.addRow(new Object[]{ appointment_id, patient_id, doctor_id, p_name, d_name, date, time, patient_info});
61         }
62     }
63     rn.close();
64     stat1.close();
65     con.close();
66 }
67 catch(Exception e)
68 {
69     System.out.println(e);
70 }
71

```

Fig 4.37: Patient's appointments code snippet

The ‘List of Appointments’ frame in the doctor’s portal contains all the information taken from the Appointment table about all the appointments that have been booked for the future by a patient.



Fig 4.38: Doctor's appointments

```

    public List_app(String username, String password, String name, int ppid, int did) {
        initComponents();
        setUsername(username);
        setPassword(password);
        setLocation(ppid);
        did(did);
        try {
            Class.forName("java.sql.DriverManager");
            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace", "root", "Ishika-gupta@5");
            Statement stmcncc.createStatement();
            String query="select * from appointment where did='"+did+"'";
            ResultSet rs=stmcncc.executeQuery(query);
            while(rs.next())
            {
                DefaultTableModel model=(DefaultTableModel) table.getModel();
                String p_name=rs.getString(1);
                String time=rs.getString(3);
                String date=rs.getString(4);
                int p_id=rs.getInt(5);
                int a_id=rs.getInt(7);
                String p_info=rs.getString(8);
                model.addRow(new Object[]{(a_id, p_id, p_name, p_info, time, date)});
            }
        } catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
/*
 * This method is called from within the constructor to initialize the form.
 */

```

Fig 4.39: Doctor's appointments code snippet

The ‘Book Appointment’ feature in the patient login portal enables the user to book an appointment with any doctor of their liking (using their doctor ID). In this frame, the user has the freedom to choose the date and time slots for the appointment and they have a textbox that lets the doctor know in-depth about the health problem they are facing. If a patient tries to book a time or date slot for a doctor that has already been booked, a message is popped up invalidating all the information provided by the user, the appointment will not be booked. Once all values are appropriate, all information provided by the user is stored in the Appointment table.

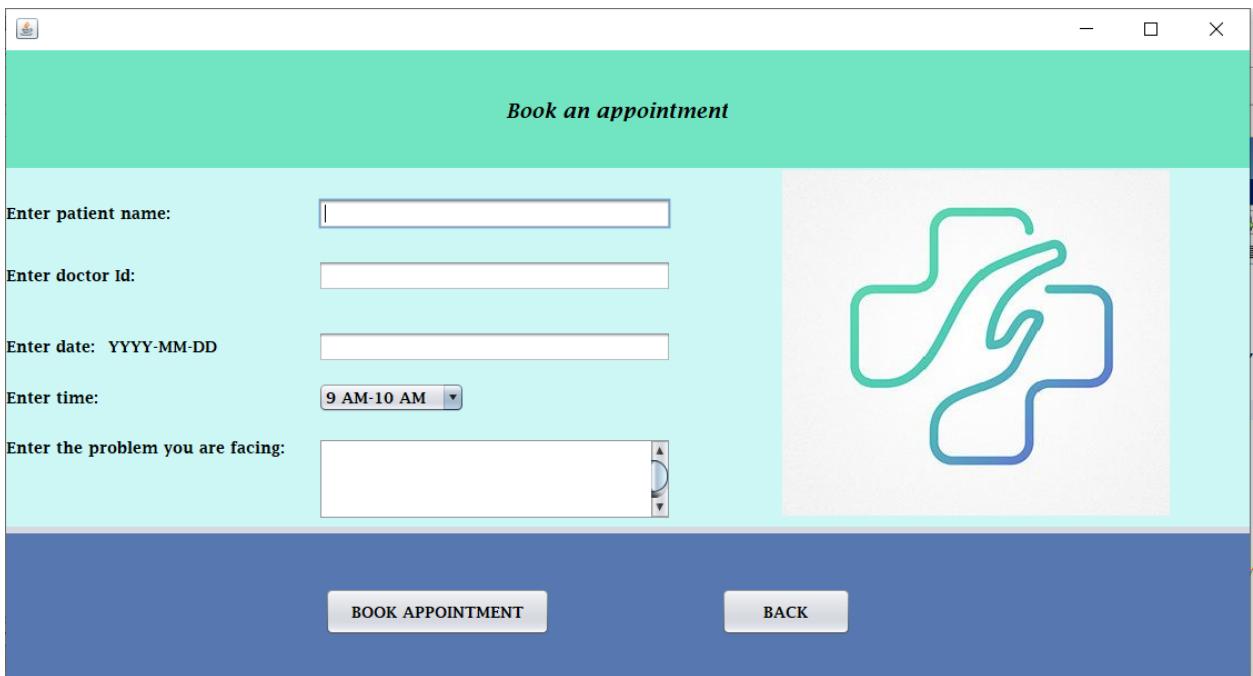


Fig 4.40: Booking an appointment

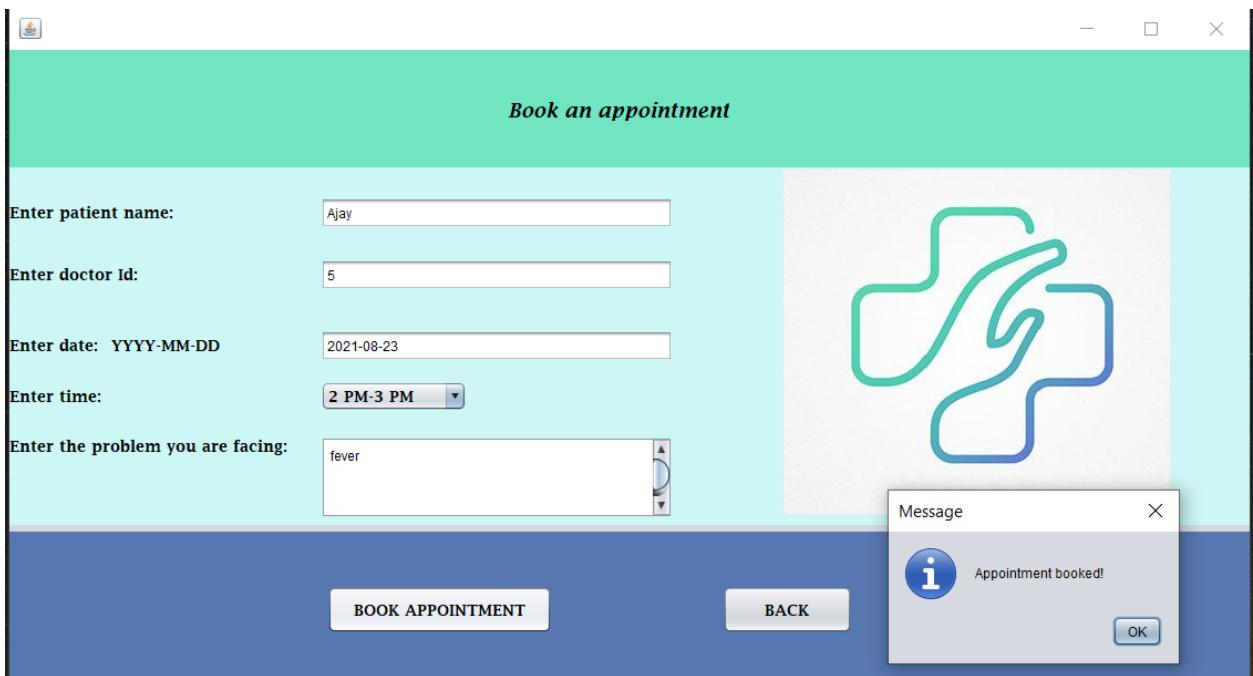


Fig 4.41: Appointment booked

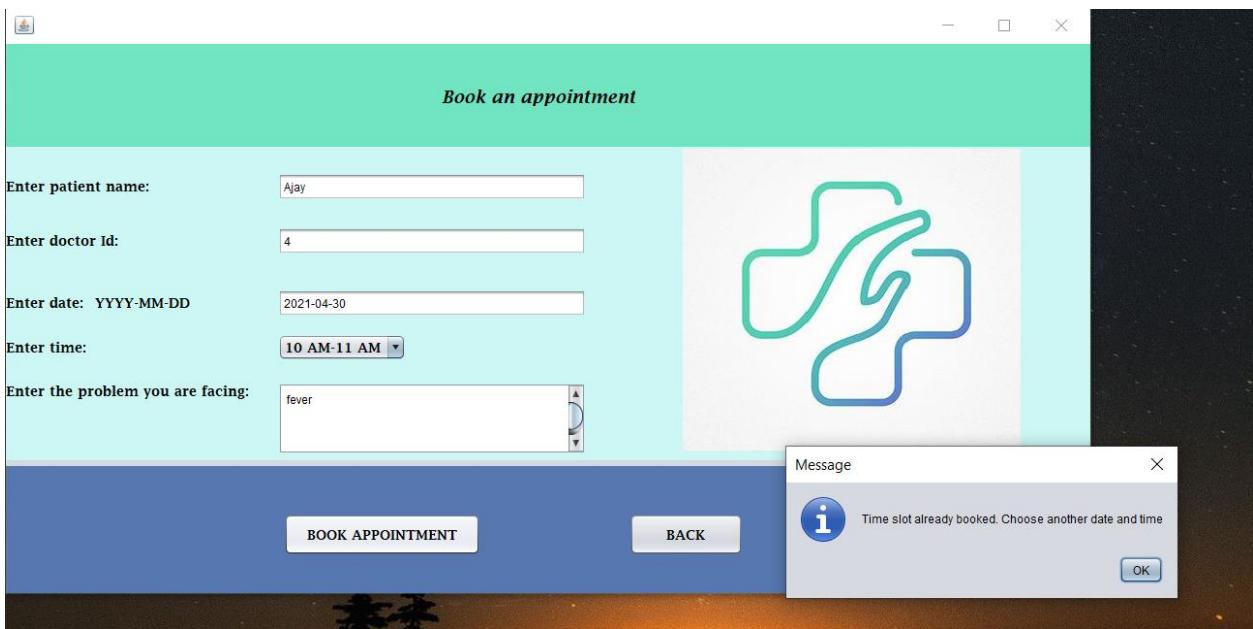


Fig 4.42: Invalid appointment booking

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        String p_name=jbn1.getText();
        int d_id=Integer.parseInt(jbn2.getText());
        String date=jbn3.getText();
        String time=(String)jcb1.getSelectedItem();
        String problem=jcb2.getSelectedItem();
        String name="";
        String dbDate=new String[100];
        String dbTime[]=new String[100];
        int numRow=0;
        try {
            Class.forName("java.sql.DriverManager");
            Connection conl=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta15");
            Statement statl=conl.createStatement();
            String query="select * from doctor where did='"+d_id+"'";
            ResultSet rs=statl.executeQuery(query);
            if(rs.next())
            {
                d_name=rs.getString(1);
                did=rs.getInt(2);
                System.out.println(did);
            }
            rs.close();
            statl.close();
            conl.close();
        } catch(Exception e)
        {
            System.out.println(e);
        }
        try
        {
            String query="insert into appointment values('"+name+"','"+p_name+"','"+date+"','"+time+"','"+problem+"','"+did+"')";
            Statement stat=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta15").createStatement();
            stat.executeUpdate(query);
            JOptionPane.showMessageDialog(null,"Appointment Booked");
        }
        catch(Exception e)
        {
            JOptionPane.showMessageDialog(null,e);
        }
    }
}

```

Fig 4.43: Appointment booking code snippet 1

The screenshot shows the NetBeans IDE interface with the following details:

- Title Bar:** Health_space - NetBeans IDE 8.2 RC
- Menu Bar:** File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
- Toolbar:** Standard NetBeans toolbar with icons for file operations.
- Project Explorer:** Shows the project structure under "Projects".
- Code Editor:** Displays Java code for `JOptionPane.java`. The code is part of a try-catch block, specifically handling a database connection and statement. It includes SQL queries for selecting and inserting data into the "appointment" table.
- Output:** Shows the output for the "Health_space (run)" configuration.
- Bottom Status Bar:** Shows the page number 48.

```

283         }
284     } catch(Exception e)
285     {
286         System.out.println(e);
287     }
288     try
289     {
290         Class.forName("java.sql.DriverManager");
291         Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","fishka-gupta@8");
292         Statement stmt=con.createStatement();
293         String query="select * from appointment where Time='"+time+"' and date='"+date+"' and did='"+d_id+"'";
294         ResultSet rs2=stmt.executeQuery(query2);
295         if(rs2.next())
296         {
297             JOptionPane.showMessageDialog(null,"Time slot already booked. Choose another date and time");
298         }
299         else
300         {
301             String query="Insert into appointment(PatientName,DoctorName,Time,Date,did,pInfo) values('"+p_name+"','"+d_name+"','"+time+"','"+date+"','"+did+"','"+p_info+"')";
302             stmt.executeUpdate(query);
303             JOptionPane.showMessageDialog(null,"Appointment booked!");
304             billing b=new billing(p_id,d_id);
305             this.setVisible(false);
306             b.setVisible(true);
307             stmt.close();
308             con.close();
309         }
310     }
311     catch(Exception e)
312     {
313         System.out.println(e);
314     }
315 }
```

Fig 4.44: Appointment booking code snippet 2

Once the appointment is booked, the user is directly navigated to the payment page where the appointment's bill is displayed. The user is asked to transfer the said amount to the given account number and provide the transaction ID of their payment. Once the transaction ID is validated by clicking the 'Pay fee online' button, a message is popped up that confirms the user's payment.

The payment changes are made in the Appointment table.

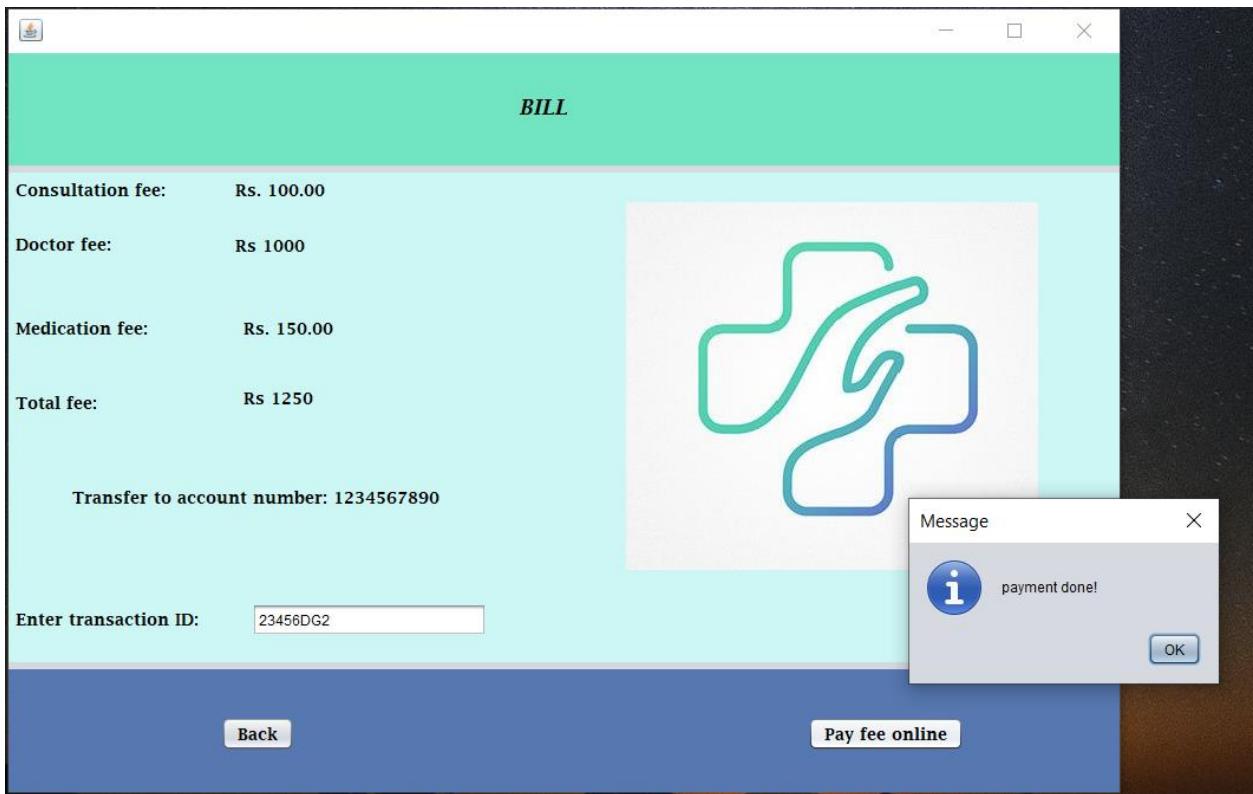


Fig 4.45: Fee payment

The 'Cancel Appointment' feature in the patient login portal enables the user to cancel any appointment they have booked due to any inconvenience. The frame asks for the appointment ID of the appointment they wish to delete which the patient can view from the 'View Your Appointments' feature. Once a valid appointment ID is added, a message is popped up that confirms the cancellation and refunds the amount that was paid during the booking of that appointment.

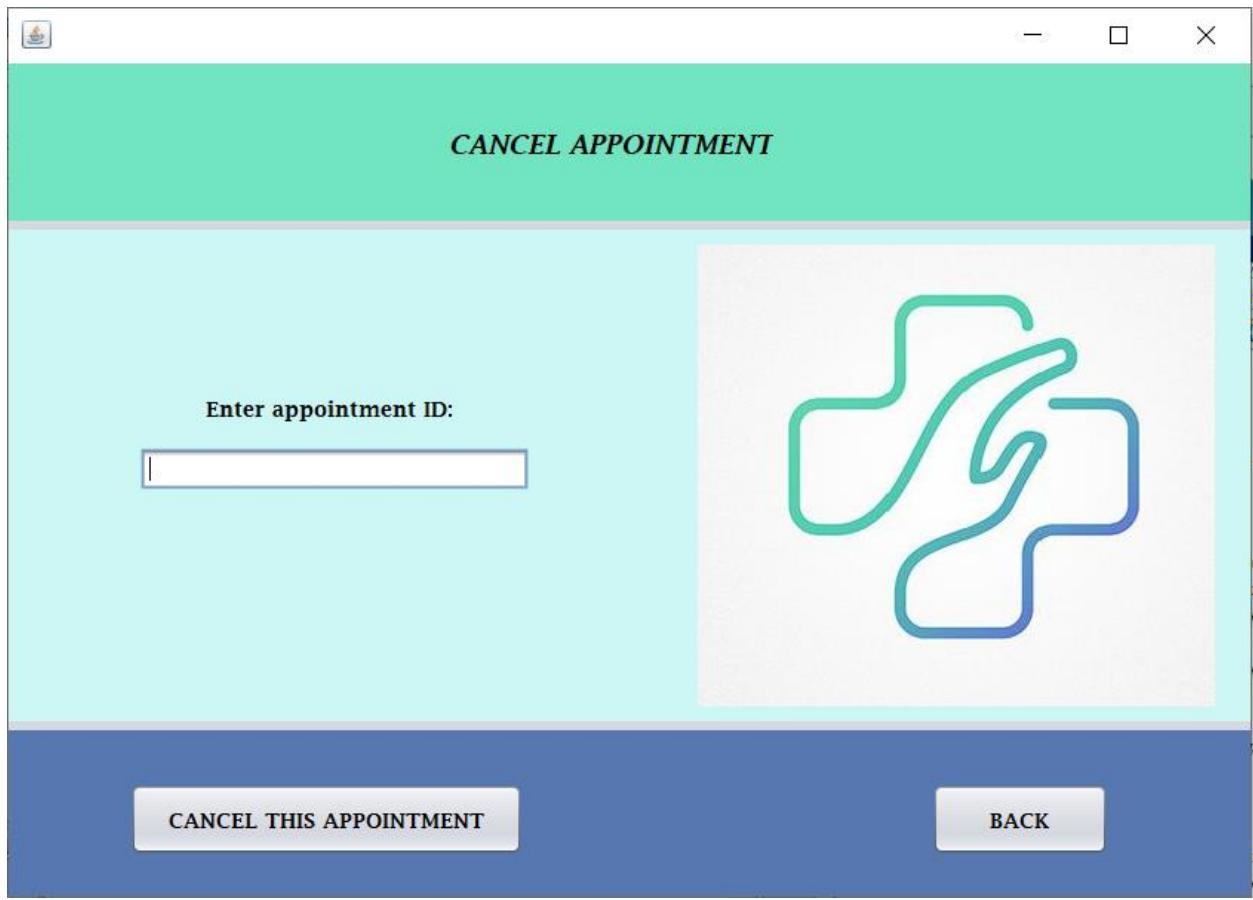


Fig 4.46: Cancelling appointment

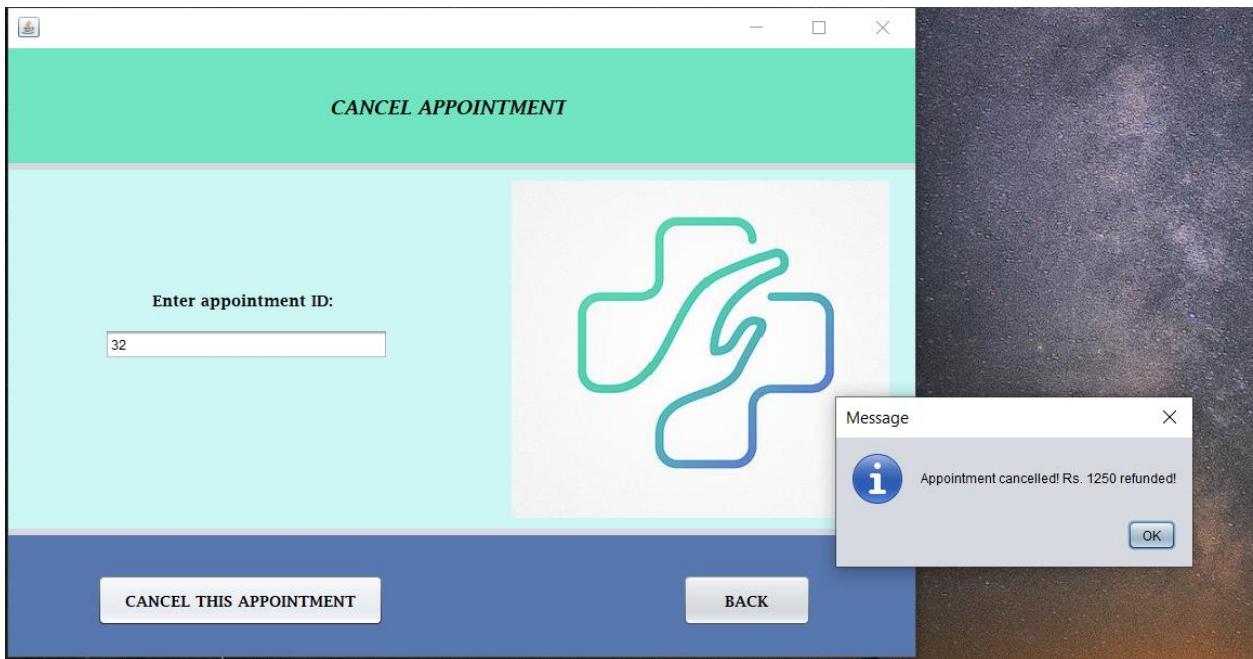


Fig 4.47: Refund after cancellation

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int aid=Integer.parseInt(cal.getText());
    String fee="";
    try {
        Class.forName("java.sql.DriverManager");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta45");
        Statement stat=con.createStatement();
        String query1="delete from appointment where aid='"+aid+"'";
        String query2="select * from appointment where aid='"+aid+"'";
        ResultSet resstatm.executeQuery(query1);
        while(rs.next())
        {
            fee=rs.getString(9);
        }
        stat.executeUpdate(query1);

        stat.close();
        con.close();
    } catch(Exception e) {
        System.out.println(e);
    }
    JOptionPane.showMessageDialog(null,"Appointment cancelled! Rs. "+fee+" refunded!");
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Patient_Login pl=new Patient_Login(n,u,p.pid,did);
    pl.setVisible(true);
    this.setVisible(false);
}

```

Fig 4.48: Cancelling appointment code snippet

Once the appointment is done, the doctor can use the ‘Write Report’ and ‘Write Prescription’ features to write down the results of all the tests that were conducted, the suggestions the doctor has for the patient and the outcomes of the appointment. The doctor can access that particular appointment using the appointment ID which the doctor can view from the ‘List of Appointments’ feature. Once the doctor writes the report and prescription, the text is stored in the Result table. This report is then projected in the ‘View Report’ and ‘View Prescription’ features of the patient.

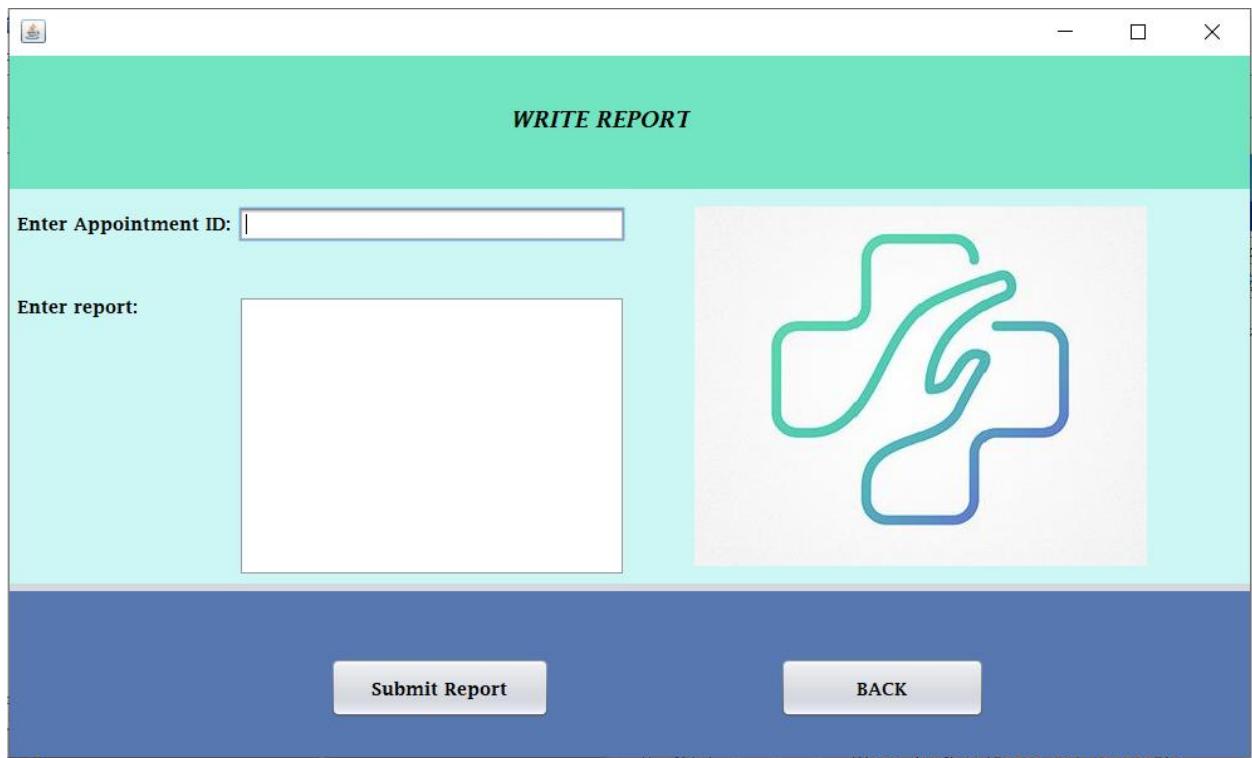


Fig 4.49: Writing report

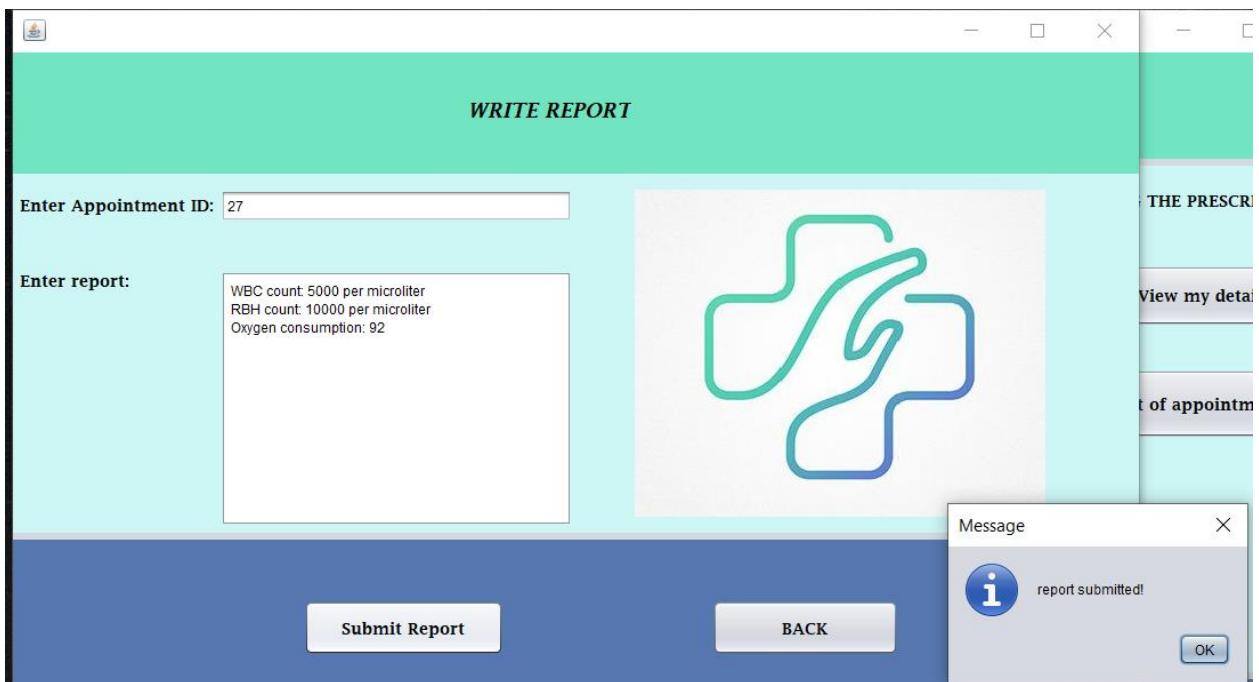


Fig 4.50: Submitting report

Health_space - NetBeans IDE 8.2 RC

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Start Page JoptionPane.java

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here
    int a_id=Integer.parseInt(t1.getText());
    String report=t2.getText();
    try {
        Class.forName("java.sql.DriverManager");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta5");
        Statement stat=con.createStatement();
        //String query="update result set report='"+report+"' Where AID='"+a_id+"'";
        String query="insert into result (AID,report) values ('"+a_id+"','"+report+"')";
        stat.executeUpdate(query);
        stat.close();
        con.Close();
    } catch(Exception e) {
        System.out.println(e);
    }
    JOptionPane.showMessageDialog(null,"report submitted!");
}

```

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here
 Doctor_Login d1=new Doctor_Login(n,u,p,did);
 d1.setVisible(true);
 this.setVisible(false);
}

Fig 4.51: Writing report code snippet

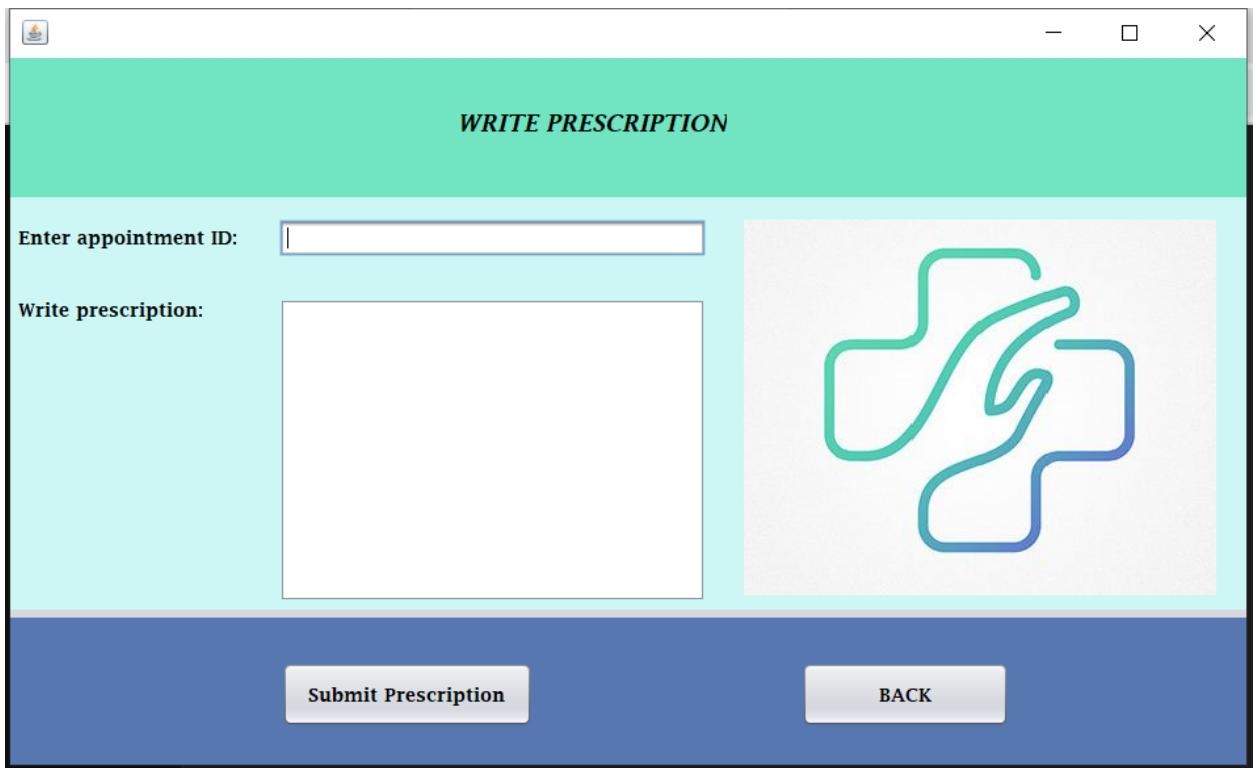


Fig 4.52: Writing prescription

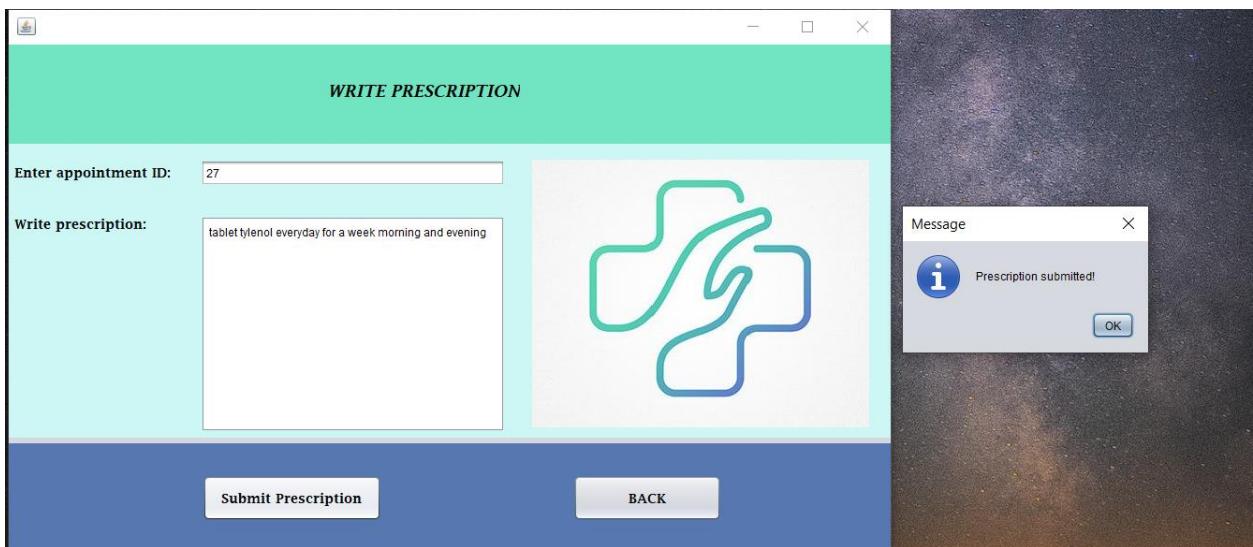


Fig 4.53: Submitting prescription

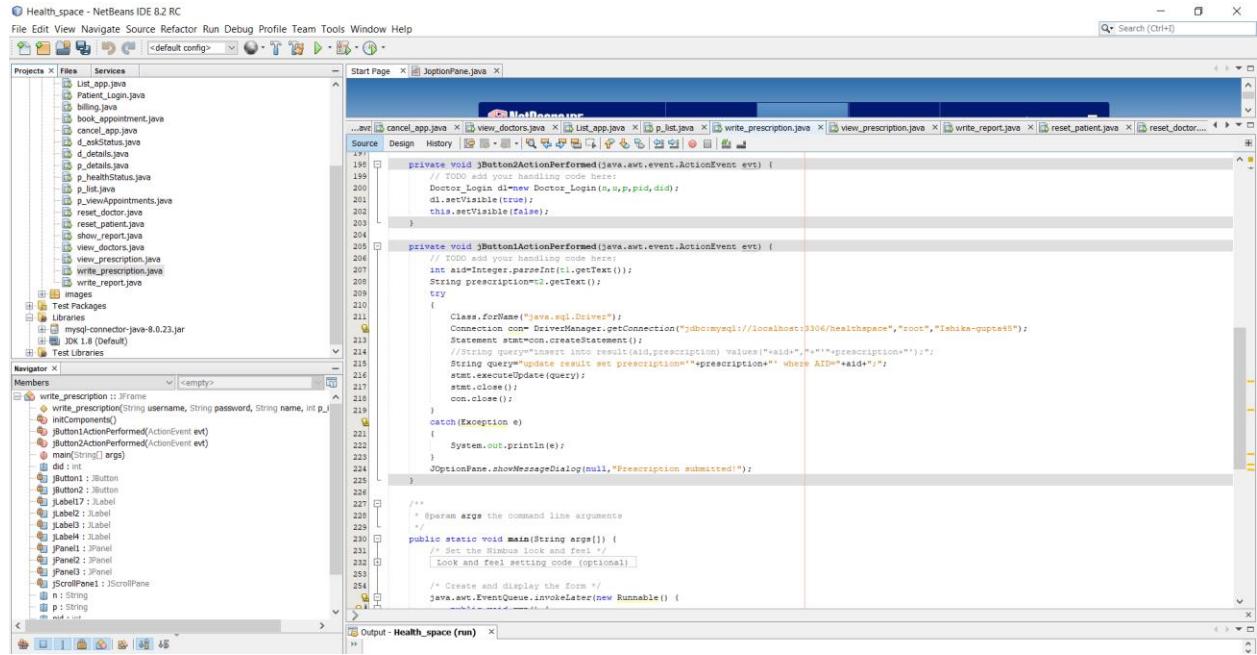


Fig 4.54: Writing prescription code snippet

The ‘View Report’ and ‘View Prescription’ features in the patient login porta enable the user to view the report and prescription written by the doctor using the ‘Write Report’ and ‘Write Prescription’ features after the appointment. This frame asks the user for the appointment ID of the appointment they wish to see the report of. Once appropriate value is added, the report is displayed from the Result table.

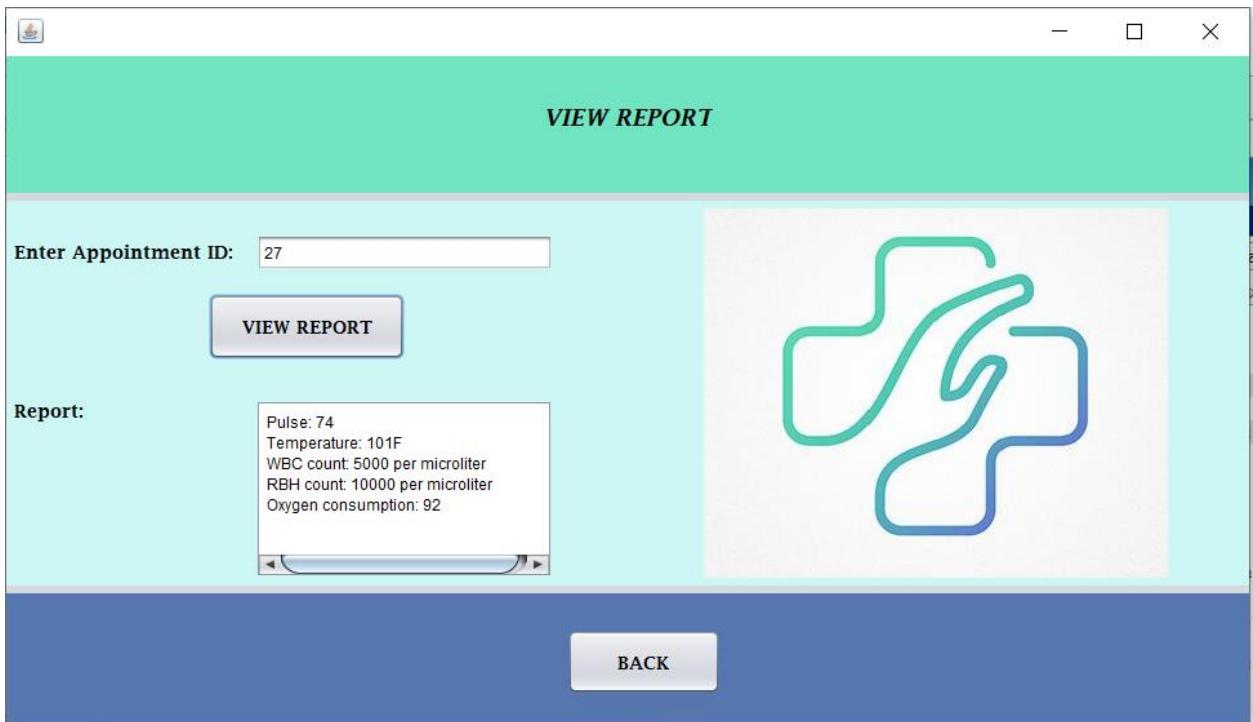


Fig 4.55: Viewing report

Health_space - NetBeans IDE 8.2 RC

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Start Page | JoptionPane.java |

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int a_id=Integer.parseInt(t1.getText());
    String report="";
    try
    {
        Class.forName("Java.sql.DriverManager");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta15");
        Statement stat=con.createStatement();
        String query="select * from result where AID='"+a_id+"'";
        ResultSet rs=stat.executeQuery(query);
        while(rs.next())
        {
            report+=rs.getString(3);
        }
        rs.close();
        stat.close();
        con.close();
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    t2.setText(report);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Patient_Login pl=new Patient_Login(n,u,p,pid,did);
    pl.setVisible(true);
    this.setVisible(false);
}

```

Fig 4.56: Viewing report code snippet

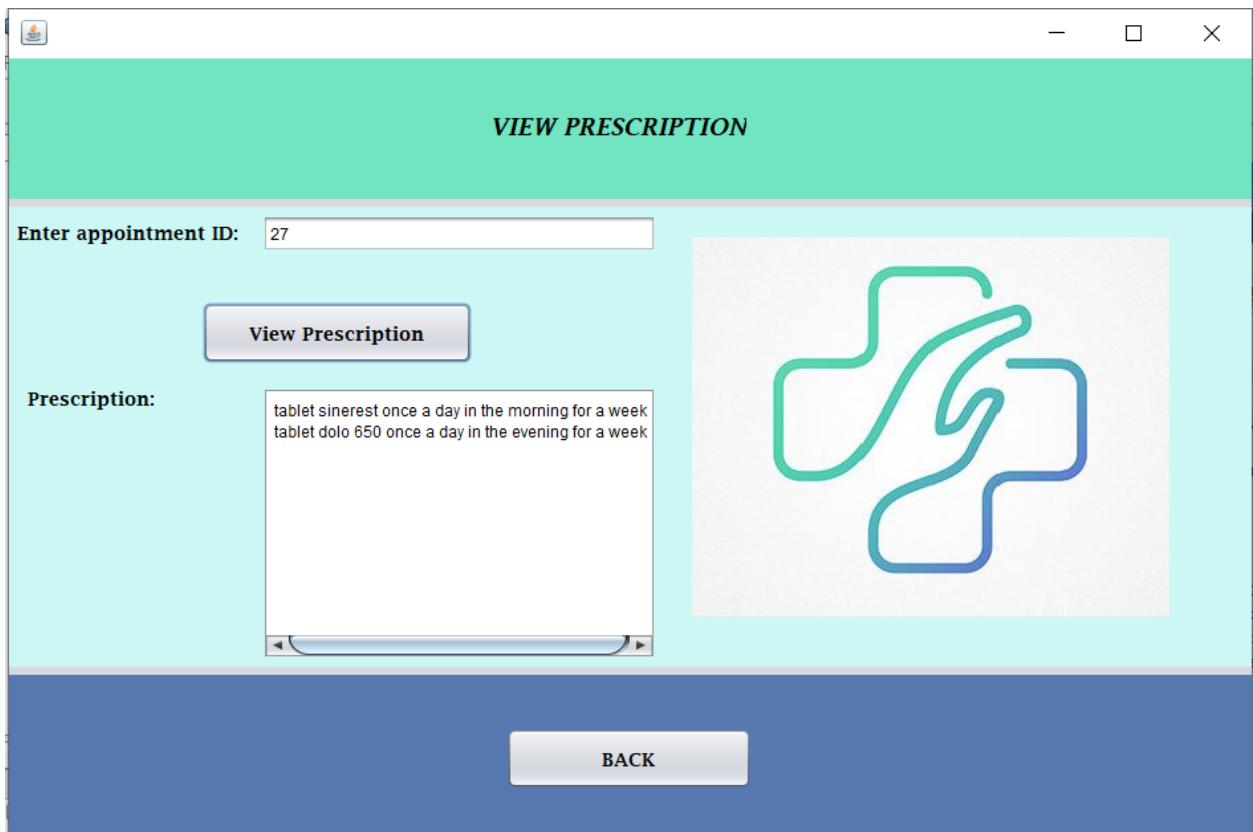


Fig 4.57: Viewing prescription

```

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    Patient_Login pl=new Patient_Login(n,u,p,plid,did);
    pl.setVisible(true);
    this.setVisible(false);
}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int aid=Integer.parseInt(t1.getText());
    String prescription="";
    try
    {
        Class.forName("java.sql.DriverManager");
        Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/healthspace","root","Ishika-gupta5");
        Statement stmt=con.createStatement();
        String query="select * from result where AID='"+aid+"'";
        ResultSet rs=stmt.executeQuery(query);

        while(rs.next())
        {
            prescription=rs.getString(2);
        }
        rs.close();
        stmt.close();
        con.close();
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
    t2.setText(prescription);
}

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
public static void main(String args[]) {
}

```

Fig 4.58: Viewing prescription code snippet

CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION:-

From the above stated facts and information, the importance of health is well-understood and that the usage of the Healthspace enables improved patient care. ‘Healthspace’ makes managing information very convenient, for the patient as well as the doctor. It is an application that will stick by its user right from the registration process and keeps constant track of any updates. With different logins for both patient and doctor, one can easily access information relevant only to them. Its value also notices a surge with the ongoing pandemic that urges to take a closer look at their health.

5.2 LIMITATIONS:-

The main restriction is that the given application is just available on desktops and not mobiles. The features are limited only to the basic appointment requirements. Also, the input given in the current database isn’t enough for complete and effective use of the application. The chat box feature is rudimentary and doesn’t enable us to retain the previous conversations between the doctor and patient.

5.3 FUTURE SCOPE:-

- We hope to move on from not just being a desktop application but to also being a mobile application for increasing its accessibility.
- We hope to provide necessary information about different resources available in different hospitals.
- We hope to improve our fundamental chat-box feature to improve the relationship between doctors and patients.
- We hope to shift our paradigm to a wider range of hospitals than just a single one.

BIBLIOGRAPHY

- Informatics Practices (Volume 1), a textbook by Sumita Arora Dhanpat Rai & Co. (Pvt.) Ltd. Educational and Technical Publishers
- Informatics Practices(Volume 2), a textbook by Sumita Arora Dhanpat Rai & Co. (Pvt.) Ltd. Educational and Technical Publishers
- https://en.wikipedia.org/wiki/Health_management_system
- <https://www.javatpoint.com/java-jdbc>